République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abderrahmane Mira
Faculté des Sciences Exactes
Département d'Informatique

Mémoire de Fin de Cycle

Pour l'obtention du diplôme de Master Recherche en Informatique

Option :
Réseaux et Sécurité

Théme

La détection d'intrusion dans un système informatique

Préparé par :

Tighidet Ferhat

Présidente	Pr. L. Bouallouche	Professeur	U. A/Mira Béjaia.
Encadrant	Pr. R.Beghdad	Professeur	U. A/Mira Béjaia.
Examinateur	Dr. M. SADI	Maître de conf. B	U. A/Mira Béjaia.
Examinatrice	Dr. N. Bouadem	Maître de conf. B	U. A/Mira Béjaia.
Examinatrice	Dr. S. Ouyahia	Maître de conf. B	U. A/Mira Béjaia.

Année universitaire : 2024/2025

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

IDS Intrusion Detection System (Système de Détection d'Intrusion)

IA Intelligence Artificielle

IPS Intrusion Prevention System (Système de Prévention d'Intrusion)

NIDS Network-based Intrusion Detection System (Système de Détection d'Intrusion Basé sur le Réseau)

HIDS Host-based Intrusion Detection System (Système de Détection d'Intrusion Basé sur l'Hôte)

NNIDS Network Node Intrusion Detection System (Système de Détection d'Intrusion Basé sur les Nœuds du Réseau)

ABIDS Application-based Intrusion Detection System (Système de Détection d'Intrusion Basé sur les Applications)

SBIDS Stack-Based Intrusion Detection System (Système de Détection d'Intrusion Basé sur la Pile)

DoS Denial of Service (Déni de Service)

DDoS Distributed Denial of Service (Déni de Service Distribué)

R2L Remote to Local (Accès à Distance vers Local)

U2R User to Root (Utilisateur vers Root)

TCP Transmission Control Protocol (Protocole de Contrôle de Transmission)

UDP User Datagram Protocol (Protocole de Datagramme Utilisateur)

ICMP Internet Control Message Protocol (Protocole de Message de Contrôle Internet)

ML Machine Learning (Apprentissage Automatique)

DL Deep Learning (Apprentissage Profond)

SMOTE Synthetic Minority Oversampling Technique (Technique de Suréchantillonnage des Minorités Syntétiques)

PCA Principal Component Analysis (Analyse en Composantes Principales)

RFE Recursive Feature Elimination (Élimination Récursive de Caractéristiques)

SVM Support Vector Machine (Machine à Vecteurs de Support)

LASSO Least Absolute Shrinkage and Selection Operator (Opérateur de Rétrécissement et de Sélection Absolus Minimum)

HTTP HyperText Transfer Protocol (Protocole de Transfert Hypertexte)

IP Internet Protocol (Protocole Internet)

SIEM Security Information and Event Management (Gestion des Informations et des Événements de Sécurité)

NIST National Institute of Standards and Technology (Institut National des Normes et de la Technologie)

DMZ Demilitarized Zone (Zone Démilitarisée)

TP True Positives (Vrais Positifs)

TN True Negatives (Vrais Négatifs)

FP False Positives (Faux Positifs)

FN False Negatives (Faux Négatifs)

TPR True Positive Rate (Taux de Vrais Positifs)

FPR False Positive Rate (Taux de Faux Positifs)

AUC Area Under the Curve (Aire sous la courbe)

CNN Convolutional Neural Network (Réseau de Neurones Convolutifs)

LSTM Long Short-Term Memory (Mémoire à Court et Long Terme)

DNN Deep Neural Network (Réseau de Neurones Profond)

RF Random Forest (Forêt Aléatoire)

DBN Deep Belief Network (Réseau de Croyance Profond)

RBM Restricted Boltzmann Machine (Machine de Boltzmann Restreinte)

MLP Multi-Layer Perceptron (Perceptron Multicouche)

IoT Internet of Things (Internet des Objets)

EDA Exploratory Data Analysis (Analyse Exploratoire des Données)

ANOVA Analysis of Variance (Analyse de la Variance)

ReLU Rectified Linear Unit (Unité Linéaire Rectifiée)

SMO Spider Monkey Optimization (Optimisation par les Singes Araignées)

BiLSTM Bidirectional Long Short-Term Memory (Mémoire à Court et Long Terme Bidirectionnelle)

SDN Software-Defined Networking

TCN Temporal Convolutional Network (Réseau Convolutionnel Temporel)

DS-kNN Distance Sum-based k-Nearest Neighbors (k-Plus Proches Voisins Basé sur la Somme des Distances)

ViT Vision Transformer (Transformeur de Vision)

HAST-IDS Hybrid Attention-based Spatio-Temporal Intrusion Detection System(architecture d'intrusion basée sur l'attention spatio-temporelle)

1D-CNN One-Dimensional Convolutional Neural Network (Réseau de Neurones Convolutifs Unidimensionnel)

ANN Artificial Neural Network (Réseau de Neurones Artificiel)

SDN Software-Defined Networking (Réseau Définis par Logiciel)

FESVDF Feature Selection using Singular Value Decomposition (Sélection de Caractéristiques par Décomposition en Valeurs Singulières)

SHIA Security Hybrid Intelligent Approach (Approche Hybride Intelligente de Sécurité)

SGM-CNN Stochastic Gradient Method with Convolutional Neural Network (Méthode de Gradient Stochastique avec Réseau de Neurones Convolutifs)

AI-SIEM Artificial Intelligence Security Information and Event Management (Gestion des Informations et Événements de Sécurité par Intelligence Artificielle)

InSDN Intrusion Detection in Software-Defined Networking (Détection d'Intrusion dans les Réseaux Définis par Logiciel)

BERT Bidirectional Encoder Representations from Transformers (Représentations de Codeurs Bidirectionnels de Transformateurs)

XAI Explainable Artificial Intelligence (Intelligence Artificielle Explicable)

LIME Local Interpretable Model-agnostic Explanations (Explications Interprétables Locales Indépendantes du Modèle)

SHAP SHapley Additive exPlanations (Explications Additives de Shapley)

HiViT-IDS Hybrid Vision Transformer for Intrusion Detection System (Transformeur de Vision Hybride pour les Systèmes de Détection d'Intrusion)

GELU Gaussian Error Linear Unit (Unité Linéaire Gaussienne Exponentielle)

ROC Receiver Operating Characteristic (Courbe Caractéristique de Fonctionnement du Récepteur)

TPU Tensor Processing Unit (Unité de Traitement Tensoriel)

GPU Graphics Processing Unit (Unité de Traitement Graphique)

CPU Central Processing Unit (Unité Centrale de Traitement)

vCPU Virtual Central Processing Unit (Unité Centrale de Traitement Virtuelle)

vRAM Video Random Access Memory (Mémoire Vidéo à Accès Aléatoire)

Remerciements

Je tiens tout d'abord à exprimer ma profonde gratitude à **Dieu Tout-Puissant**, qui m'a accordé la santé, la patience et la force nécessaires pour mener à bien ce travail.

Je remercie sincèrement mon encadrant, **Pr. Beghdad Rachid**, pour sa disponibilité, ses conseils avisés, son suivi rigoureux ainsi que pour la confiance qu'il m'a accordée tout au long de ce projet. Son encadrement m'a permis de progresser tant sur le plan académique que personnel.

Je souhaite également adresser mes remerciements les plus respectueux aux **membres du jury** pour l'honneur qu'ils me font en acceptant d'évaluer ce mémoire. Leur lecture attentive et leurs remarques constructives sont grandement appréciées.

À tous ceux qui, de près ou de loin, ont contribué à la réalisation de ce mémoire, je vous adresse mes plus sincères remerciements.

Dédicace

Je dédie ce travail, fruit de plusieurs années d'efforts et de persévérance :

À mon père et à ma mère, pour leur amour inconditionnel, leur soutien indéfectible et leurs innombrables sacrifices. Je ne pourrai jamais les remercier à leur juste valeur. Que Dieu les protège et les récompense pour tout ce qu'ils ont fait pour moi.

À mes sœurs, Dania et Melissa, pour leur présence, leur affection et leur encouragement tout au long de mon parcours.

À mes grands-parents, du côté paternel et maternel, que Dieu les garde et les préserve encore longtemps. Leur sagesse, leur bienveillance et leurs prières m'ont toujours accompagné.

À mes oncles et mes tantes, ainsi qu'à leurs enfants, que la liste est longue pour tous les citer, mais que chacun se reconnaisse dans cette pensée pleine de gratitude.

À tous mes amis.

Et enfin, à tous ceux que j'aime et qui m'aiment d'un amour sincère.

Merci à vous tous, ce mémoire vous est dédié.

Introduction Générale

La détection d'intrusion est devenue une problématique cruciale dans le domaine de la cybersécurité, en raison de l'augmentation exponentielle des menaces numériques et de la sophistication des attaques ciblant les réseaux informatiques. Les systèmes traditionnels, basés sur des règles prédéfinies ou des signatures, peinent à identifier les nouvelles formes d'attaques ou les comportements anormaux non répertoriés, exposant ainsi les infrastructures critiques à des risques majeurs. Cette limitation soulève la nécessité d'adopter des approches plus adaptatives et capables d'analyser de grands volumes de données en temps réel, posant ainsi les bases pour l'exploration de solutions avancées telles que l'apprentissage profond.

Avec l'augmentation constante de la complexité des réseaux et du volume de données à analyser, les approches traditionnelles de détection se révèlent souvent insuffisantes face aux menaces émergentes ou rares. Afin de surmonter ces limites, la communauté scientifique s'oriente de plus en plus vers des approches basées sur l'apprentissage profond, capables d'extraire automatiquement des caractéristiques pertinentes et de mieux s'adapter à des contextes variés. Les réseaux de neurones convolutifs (CNN), les réseaux neuronaux profonds (DNN) et les modèles récurrents comme les LSTM (Pour : *Long Short-Term Memory*) ont déjà démontré leur efficacité, notamment en matière de précision — c'est-à-dire leur capacité à identifier correctement les attaques sans générer trop de faux positifs — et de généralisation, soit leur aptitude à maintenir de bonnes performances sur des données inédites. Les CNN (Convolutional Neural Network) sont particulièrement adaptés à la détection de motifs locaux, tels que des combinaisons spécifiques de caractéristiques dans un court laps de temps ou dans une séquence de paquets réseau, tandis que les LSTM permettent de modéliser les dépendances temporelles du trafic, en capturant les relations entre événements espacés dans le temps.

Récemment, les Transformers ont été introduits dans le domaine de la détection d'intrusion, notamment via leur adaptation aux tâches de vision par ordinateur. Le Vision Transformer (ViT Pour : *Vision Transformer*), en appliquant une attention globale à l'ensemble de l'image, offre de bonnes performances en matière d'exactitude, de rappel et de F1-score dans la classification d'attaques réseau, mais au prix d'un coût de calcul élevé. Pour pallier cette contrainte, le Swin Transformer (Shifted Window Transformer) propose une attention locale hiérarchique via des fenêtres glissantes, permettant de maintenir des performances élevées — notamment une précision comparable à celle du ViT — tout en réduisant significativement la complexité de calcul. Cette architecture s'avère donc prometteuse pour des applications à grande échelle, notamment dans les environnements de cybersécurité.

Dans ce mémoire, nous proposons une approche hybride combinant les CNN et le Swin Transformer pour la détection d'intrusion dans les réseaux informatiques. Afin d'exploiter les capacités de ces architectures de vision, nous transformons les données tabulaires issues des ensembles de données NSL-KDD et CIC-IDS-2017 en images en niveaux de gris. Notre processus d'expérimentation est structuré en

quatre phases : (1) La préparation et la normalisation des données, puis (2) La transformation en images, ensuite, (3) Entraînement du modèle hybride CNN–Swin Transformer, enfin (4) Évaluation sur des tâches de classification binaire et multiclasse.

Les résultats obtenus montrent que notre approche offre des performances supérieures à celles de certaines méthodes existantes, atteignant jusqu'à 99,69 % d'exactitude en classification binaire sur NSL-KDD et 98,28 % sur CIC-IDS-2017, tout en assurant une meilleure robustesse pour les classes minoritaires, notamment en améliorant la détection des attaques rares telles que R2L et U2R, et ce, sans recourir à des techniques de rééchantillonnage ou de sélection manuelle de caractéristiques. En classification multiclasse, elle atteint 98,17 % d'exactitude sur NSL-KDD et 96,87 % sur CIC-IDS-2017, confirmant ainsi la pertinence de cette fusion CNN–Swin Transformer pour renforcer les performances des IDS (Pour *Intrusion Detection System*) modernes.

Le présent mémoire est structuré comme suit :

- Le **Chapitre I** introduit les systèmes de détection d'intrusion, leurs différentes classifications, ainsi que les concepts fondamentaux de l'intelligence artificielle appliqués à la cybersécurité.
- Le **Chapitre II** présente un état de l'art des approches existantes basées sur l'apprentissage automatique et profond, à travers une revue critique des travaux récents.
- Le **Chapitre III** décrit l'approche proposée, en détaillant les modèles utilisés (CNN et Swin Transformer), la méthodologie adoptée et les étapes du processus.
- Le **Chapitre IV** expose les résultats expérimentaux sur les jeux de données NSL-KDD et CIC-IDS-2017, en les comparant aux approches de l'état de l'art.
- Enfin, la **Conclusion** qui conclut ce mémoire et propose des perspectives pour des travaux futurs dans le domaine de la détection d'intrusion intelligente.

Chapitre 1
Concepts et Méthodes de Détection des Intrusions et Applications de l'IA en Cybersécurité

1.1 Introduction

Dans un contexte où les cyberattaques deviennent de plus en plus sophistiquées et fréquentes, la sécurisation des systèmes d'information est devenue une priorité incontournable pour les organisations. Les systèmes de détection d'intrusion (IDS) jouent un rôle central dans cette démarche, en surveillant les réseaux et les systèmes pour identifier et signaler les activités malveillantes. Ce chapitre explore les concepts fondamentaux et les méthodes associées aux systèmes de détection d'intrusion, ainsi que l'application de l'intelligence artificielle (IA Pour : *Intelligence Artificielle*) dans ce domaine.

Nous commencerons par une présentation des IDS, en détaillant leur définition, leur rôle et les différentes classifications possibles. Nous aborderons également les limites des systèmes classiques et l'importance de l'IA pour surmonter ces défis. L'IA, avec ses capacités d'apprentissage automatique et d'analyse de données, offre des solutions innovantes pour améliorer la détection des intrusions. Nous examinerons les concepts clés de l'IA, les techniques de prétraitement des données, les jeux de données utilisés, ainsi que les métriques de performance pour évaluer l'efficacité des modèles de détection.

Nous discuterons de la collecte de données, un élément essentiel pour comprendre le fonctionnement et l'efficacité des systèmes de détection d'intrusion. Cet aspect est crucial pour fournir une vue d'ensemble des mécanismes de protection des systèmes informatiques. Ce chapitre vise à offrir une compréhension approfondie des méthodes et des technologies utilisées pour protéger les systèmes informatiques contre les intrusions, tout en mettant en lumière les avancées récentes et les défis futurs.

1.2 Les systèmes de détection d'intrusion (IDS)

1.2.1 Définition et rôle des systèmes de détection d'intrusion

Un IDS est un dispositif logiciel ou matériel surveillant le trafic réseau ou les événements système pour détecter des comportements malveillants, comme les accès non autorisés ou les dénis de service [1, 2]. Il protège la confidentialité (accès restreint), l'intégrité (protection des données), la disponibilité (accès continu),

l'authenticité (vérification des identités), la traçabilité (suivi des actions), et la non-répudiation (preuve des actions). Par exemple, un IDS peut signaler une tentative d'accès non autorisé à une base de données.

1.2.2 Difference entre un IDS et un Système de Prévention d'Intrusion (IPS)

Définition d'un Système de Prévention d'Intrusion

Un système de prévention d'intrusion (IPS Pour : *Intrusion Prevention System*) est une technologie de sécurité réseau qui surveille en temps réel le trafic réseau pour détecter et bloquer activement les menaces. Déployé en ligne, un IPS analyse et filtre les paquets en temps réel [3].

Différence entre un IDS et un IPS

Bien que les systèmes de détection d'intrusion (IDS) et les systèmes de prévention d'intrusion (IPS) aient des objectifs similaires en matière de cybersécurité, leur rôle et leur comportement diffèrent fondamentalement.

Un IDS a un rôle *passif* dans le sens ou il surveille le trafic réseau ou les activités système à la recherche de comportements suspects ou de signatures d'attaques connues. Lorsqu'une anomalie est détectée, l'IDS génère une alerte à destination des administrateurs systèmes ou sécurité, qui décident manuellement des mesures à prendre. Il ne prend donc pas de décision automatique de blocage ou d'intervention.

À l'inverse, un IPS agit de manière *active*. Il ne se contente pas de détecter les menaces, mais peut également les bloquer en temps réel. Cela peut inclure le rejet de paquets malveillants, la fermeture de connexions réseau suspectes ou le confinement d'un hôte compromis. L'IPS est donc capable de répondre immédiatement aux intrusions potentielles sans intervention humaine.

En résumé, l'*IDS alerte*, tandis que l'*IPS intervient*.

1.2.3 Classification des systèmes de détection d'intrusion

Les systèmes de détection d'intrusion (IDS) peuvent être classés selon plusieurs critères. Deux des plus courants sont la méthode de surveillance utilisée et la zone de surveillance ciblée [4].

Selon la méthode de surveillance

La classification suivante des systèmes de détection d'intrusion est basée sur la **méthode de détection utilisée**. Le **tableau 1.1** présente les principales méthodes, leurs avantages et leurs limites [4].

Méthode	Définition	Avantages	Limites
Détection par signature	Compare les événements aux signatures d'attaques connues.	Faible taux de faux positifs. Efficace pour les attaques connues.	Inefficace contre les attaques inconnues. Nécessite des mises à jour constantes.
Détection d'anomalie	Repère les déviations par rapport au comportement normal du système.	Capable de détecter des attaques inconnues.	Taux de faux positifs élevé. Difficulté de modélisation du comportement normal.
Détection par spécification	Utilise des règles prédéfinies par des experts pour identifier des comportements non autorisés.	Détection précise sans apprentissage préalable.	Mise en œuvre complexe à grande échelle.
Détection hybride	Combine plusieurs techniques (ex. signature et anomalie).	Meilleure couverture et robustesse.	Complexité d'intégration et besoin de ressources.

TABLE 1.1 – Classification des IDS selon la méthode de surveillance

Selon la zone de surveillance

Le **tableau 1.2** ci-dessous présente les différents types d'IDS classés selon la **zone qu'ils surveillent**, qu'il s'agisse du réseau, de l'hôte, d'une application ou d'un environnement simulé [4].

Type d'IDS	Définition	Avantages	Limites
IDS basé sur le réseau (NIDS pour *Network-based IDS*)	Surveille le trafic réseau à un point stratégique.	Non intrusif. Couvre plusieurs hôtes.	Inefficace sur trafic chiffré. Dépend du positionnement dans le réseau.
IDS basé sur l'hôte (HIDS pour *Host-based IDS*)	Analyse les événements internes d'un hôte (logs, fichiers, processus).	Précis. Accès aux données locales et chiffrées.	Vulnérable à l'attaque directe. Consomme des ressources.
IDS de nœud réseau (NNIDS pour *Network Node IDS*)	Surveille uniquement le trafic réseau destiné à un hôte particulier.	Léger, rapide. Efficace sur trafic chiffré.	Doit être installé sur chaque hôte critique.
IDS basé sur l'application (ABIDS pour *Application-based IDS*)	Analyse les interactions entre utilisateur et application.	Spécifique. Travaille en clair.	Visibilité limitée. Facile à contourner.
IDS basé sur la pile (SBIDS pour *Stack-Based IDS*)	Intégré dans la pile TCP/IP pour surveiller les paquets.	Détection rapide. Interception précoce.	Mise en œuvre complexe. Dépend du système.
Pots de miel (*Honeypots*)	Systèmes leurres visant à attirer les attaquants.	Excellente observation des attaques.	Risques légaux et d'exploitation.
Systèmes capitonnés	Simule un environnement pour piéger les intrus.	Permet d'analyser un attaquant en profondeur.	Difficulté de déploiement. Légalité discutable.

TABLE 1.2 – Classification des IDS selon la zone de surveillance

Malgré la diversité des types d'IDS présentés dans le tableau 1.2, les deux catégories les plus largement utilisées et les plus fondamentales restent :

- Les **IDS basés sur le réseau (NIDS)**, qui permettent une surveillance globale du trafic circulant sur le réseau.
- Les **IDS basés sur l'hôte (HIDS)**, qui assurent une surveillance fine et locale sur les machines critiques.

Dans les architectures modernes, il est courant d'adopter une **approche hybride** combinant les deux, afin de bénéficier à la fois d'une vision globale du trafic et d'une analyse détaillée au niveau des hôtes. Cette combinaison améliore significativement la capacité de détection et réduit les angles morts.

1.2.4 Relation entre l'IDS et le pare-feu

Les systèmes de détection d'intrusion (IDS) et les pare-feu sont deux composantes clés de la sécurité informatique, mais ils jouent des rôles distincts et complé-

mentaires dans la protection des réseaux. un pare-feu est une technologie de filtrage qui contrôle le trafic réseau en fonction de règles prédéfinies, telles que les adresses IP, les ports ou les protocoles, pour bloquer ou autoriser les connexions.[5] En revanche, un IDS se concentre sur la surveillance et l'analyse approfondie du trafic pour identifier les comportements suspects ou malveillants qui pourraient contourner les règles statiques d'un pare-feu, comme les attaques exploitant des vulnérabilités spécifiques ou des comportements anormaux [2, 5, 6, 7]. Leur complémentarité renforce la sécurité.

1.2.5 Positionnement d'un IDS dans une architecture reseau

Définition d'une zone démilitarisée (DMZ)

D'apres [8] une zone démilitarisée (en anglais DMZ pour *demilitarized zone*) est un sous-réseau séparé du réseau local et isolé de celui-ci ainsi que d'Internet (ou d'un autre réseau) par un pare-feu. Ce sous-réseau contient les machines étant susceptibles d'être accédées depuis Internet, et qui n'ont pas besoin d'accéder au réseau local.

La figure 1.1 présente une topologie réseau typique intégrant une zone démilitarisée (DMZ) à l'aide d'un pare-feu. Ce modèle permet d'isoler les serveurs accessibles depuis Internet tout en sécurisant le réseau local.

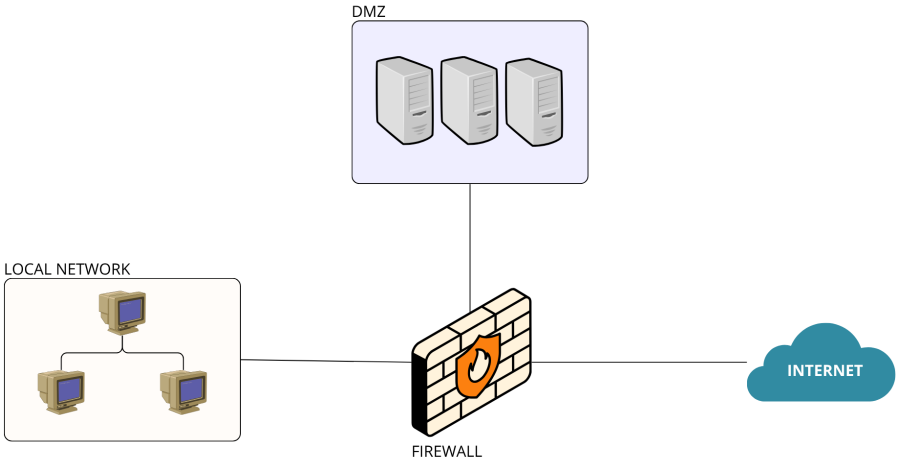


FIGURE 1.1 – Topologie réseau avec DMZ et pare-feu.

Positionnement d'un IDS

Le positionnement d'un IDS dépend du type de surveillance souhaité et de la topologie du réseau. Dans une architecture réseau intégrant une zone démilitarisée (DMZ) et un pare-feu, le placement stratégique d'un système de détection d'intrusion (IDS) est crucial pour assurer une surveillance efficace des menaces tout en optimisant les performances du réseau [9]. Cette section explore les différentes options de positionnement d'un IDS dans une telle topologie, leurs avantages et inconvénients, ainsi que l'approche hybride.

Positionnement	Avantages	Inconvénients
IDS avant le pare-feu [9]	Détection précoce des attaques (scans, DDoS). Visibilité totale sur le trafic brut. Identification des menaces avant filtrage.	Analyse de gros volumes de trafic, incluant du bruit. Nécessite un matériel performant. Moins efficace pour les menaces internes ou DMZ.
IDS après le pare-feu (DMZ) [9]	Ciblage des attaques visant la DMZ (web, mail, DNS). Moins de trafic grâce au filtrage. Détection des exploits sur services exposés.	Ne détecte pas les attaques bloquées. Limité à la DMZ, non au réseau local. Configuration complexe pour éviter les interférences.
HIDS sur chaque serveur [10]	Surveillance granulaire des serveurs (modifications, malwares). Efficace contre attaques échappant au NIDS. Idéal pour serveurs critiques.	Configuration et maintenance complexes. Ne surveille pas le trafic réseau. Consomme des ressources serveur.
Architecture hybride [9, 10]	Couverture complète (externes/internes). Flexible pour différentes topologies. Réduit les points aveugles via NIDS/HIDS.	Gestion complexe. Coût élevé en matériel et maintenance. Nécessite un SIEM pour corréler les alertes.

TABLE 1.3 – Comparaison des positionnements des systèmes de détection d'intrusion (IDS)

Pour illustrer concrètement cette approche hybride, un schéma réseau est proposé, intégrant les positions des différents systèmes de détection d'intrusion (NIDS et HIDS) dans une architecture avec un pare-feu. La figure 1.2 illustre une architecture hybride avec un pare-feu et la disposition des systèmes IDS (NIDS et HIDS) dans la DMZ et le réseau local.

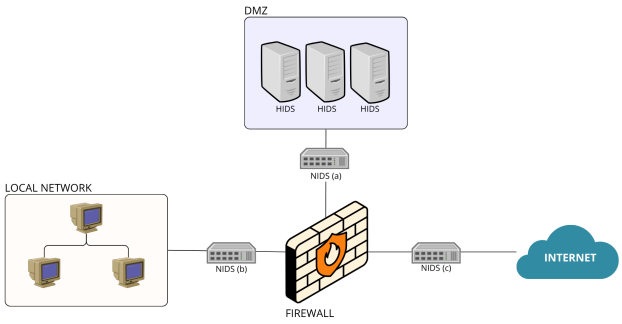


FIGURE 1.2 – Schéma d'une architecture hybride avec un pare-feu.

1.2.6 Limites des IDS classiques

Le tableau 1.1 présenté précédemment offre un aperçu synthétique des principales méthodes de détection utilisées dans les systèmes de détection d'intrusion (IDS), ainsi que de leurs avantages et inconvénients. Cependant, pour mieux comprendre leurs limites, il est pertinent de se concentrer sur les deux approches les plus répandues dans la littérature : la détection par signature et la détection par anomalie [11, 12, 13].

Limites des IDS basés sur les signatures

Les IDS basés sur les signatures comparent les événements observés à une base de données de modèles d'attaques connus. Cette approche est très fiable pour identifier des intrusions déjà répertoriées, et elle présente un faible taux de faux positifs. Toutefois, elle se révèle inefficace face aux attaques inconnues ou dites *zero-day*, en raison de sa dépendance à une base de signatures constamment mise à jour. En l'absence d'une signature adéquate, une nouvelle attaque passe inaperçue, ce qui constitue une faille majeure de cette méthode [11].

Limites des IDS basés sur les anomalies

Les IDS basés sur les anomalies, quant à eux, reposent sur l'analyse des écarts par rapport à un comportement normal appris au préalable. Ce type de système est capable de détecter des comportements malveillants non connus, ce qui en fait une méthode adaptée pour identifier des menaces émergentes. Toutefois, sa mise en œuvre est plus complexe, car elle nécessite des données d'entraînement de haute qualité et représentatives. De plus, elle présente un taux de faux positifs élevé, car toute activité inhabituelle, même légitime, peut être considérée comme suspecte. Cela peut entraîner une surcharge d'alertes et une baisse de la confiance dans le système de détection [12, 13].

Difficultés des IDS dans le traitement de grandes quantités de données

Avec l'augmentation massive du volume de données circulant dans les réseaux modernes, les IDS rencontrent des difficultés à traiter ces flux en temps réel. Le traitement efficace de grandes quantités de données nécessite des algorithmes à la fois rapides et précis, capables d'extraire des caractéristiques pertinentes sans introduire de latence importante. Les méthodes classiques peinent souvent à suivre la cadence, ce qui peut entraîner des retards de détection, des pertes de paquets, ou même une dégradation générale des performances du système de sécurité. Cela pose un réel défi, notamment dans les environnements à haut débit ou les infrastructures critiques.

1.3 Bases de l'IA pour la Sécurité Informatique

1.3.1 Concepts Fondamentaux de l'IA

Définition

L'intelligence artificielle (IA) désigne un domaine de l'informatique visant à créer des systèmes capables de simuler des comportements intelligents, tels que le raisonnement ou la reconnaissance de motifs [14]. En cybersécurité, l'IA est utilisée pour analyser le trafic réseau, détecter des menaces, prédire les attaques, classifier les activités et automatiser les réponses pour renforcer la protection.

Concepts Clés

- **Apprentissage automatique** (en anglais *machine learning*, ML) : Une sous-discipline de l'IA, permettant aux systèmes d'apprendre à partir de données sans programmation explicite. [15] définit l'apprentissage automatique comme un ensemble de méthodes améliorant les performances avec l'exposition aux données.
- **Apprentissage profond** (en anglais *deep learning*, DL) : Une branche du ML, utilisant des réseaux de neurones artificiels pour modéliser des relations complexes, comme les signatures d'attaques [16, 17].
- **Apprentissage supervisé** (en anglais *supervised learning*) : Entraîne des modèles sur des données étiquetées pour classer les intrusions (par exemple, dans NSL-KDD) [18].
- **Apprentissage non supervisé** (en anglais *unsupervised learning*) : Sans étiquettes, identifie des anomalies pour détecter des attaques inconnues (*zero-day*) [19].

1.3.2 Prétraitement des données

Le prétraitement des données constitue une étape cruciale dans l'analyse des données, visant à transformer les données brutes en un format structuré, propre et adapté à l'analyse ou à l'entraînement de modèles d'apprentissage automatique [20]. Cette étape garantit la qualité des données et améliore les performances des algorithmes [21]. Les principales techniques de prétraitement incluent :

- **Nettoyage des données :** Suppression ou imputation des valeurs manquantes, élimination des doublons et correction des incohérences (par exemple, erreurs de saisie ou formats non uniformes) pour garantir la cohérence des données.
- **Normalisation/Standardisation :** Transformation des caractéristiques numériques pour les ramener à une échelle commune, comme l'intervalle [0,1] pour la normalisation ou une distribution standard (moyenne 0, écart-type 1) pour la standardisation, afin de faciliter la convergence des modèles.
- **Encodage des variables catégoriques :** Conversion des caractéristiques non numériques en représentations numériques. Par exemple, pour les protocoles réseau (TCP, UDP, ICMP) :

  - *One-hot encoding* : Chaque catégorie est transformée en une colonne binaire. Exemple : TCP = [1, 0, 0], UDP = [0, 1, 0], ICMP = [0, 0, 1].
  - *Encodage ordinal* : Chaque catégorie est assignée à un entier selon un ordre. Exemple : TCP = 1, UDP = 2, ICMP = 3.

  Le choix dépend de la nature des données (ordonnées ou non).
- **Gestion du déséquilibre des données :** Application de techniques comme SMOTE (*Synthetic Minority Oversampling Technique*) pour générer des échantillons synthétiques des classes minoritaires, ou sous-échantillonnage des classes majoritaires, afin d'équilibrer la distribution des classes et d'éviter les biais dans les modèles.
- **Transformation des données en images ou matrices :** Pour les modèles basés sur les architectures de type *Transformers* (par exemple, dans l'analyse de séries temporelles ou de données tabulaires), les données peuvent être transformées en représentations bidimensionnelles, comme des images ou des matrices. Par exemple, les séries temporelles peuvent être converties en spectrogrammes ou en matrices de corrélation, tandis que les données tabulaires peuvent être réorganisées en embeddings adaptés aux mécanismes d'attention des *Transformers*.
- **Découpage en ensembles d'entraînement, de validation et de test :** Avant l'entraînement du modèle, le dataset est généralement divisé en plusieurs sous-ensembles :
  - *Ensemble d'entraînement* : utilisé pour ajuster les paramètres internes du modèle ;
  - *Ensemble de validation* : utilisé pour évaluer le modèle pendant l'entraînement, régler les hyperparamètres et éviter le surapprentissage (overfitting) ;
  - *Ensemble de test* : utilisé uniquement à la fin pour évaluer les performances finales sur des données totalement inconnues.

  Typiquement, on adopte une répartition de 60–70% pour l'entraînement, 15–20% pour la validation, et 15–20% pour le test.
- **Stratification :** Lors du découpage, la stratification consiste à conserver la même proportion de chaque classe dans les ensembles d'entraînement, de validation et de test. Cette technique est particulièrement utile en cas de classes déséquilibrées, car elle garantit que chaque sous-ensemble reflète fidèlement la distribution initiale des classes, réduisant ainsi les biais d'évaluation [22].

Ces étapes, adaptées au contexte et au type de données, garantissent une préparation robuste pour l'analyse ou l'entraînement de modèles, en particulier pour des architectures avancées comme les *Transformers* [21].

1.3.3 Jeux de données utilisés

Définition d'un jeux de données

Un jeux de données (*Dataset*) en apprentissage automatique est une collection de données organisées, souvent structurées en ensembles d'entraînement, de validation et de test, utilisées pour développer et évaluer des modèles prédictifs. Il

comprend des caractéristiques d'entrée (*features*) et, dans le cas de l'apprentissage supervisé, des étiquettes de sortie (*labels*), permettant l'entraînement et la validation des performances des modèles [23].

Jeux de données utilisés dans les IDS

Le tableau suivant résume les datasets les plus couramment utilisés dans ce domaine :

Datasets	**Description**
NSL-KDD [24]	NSL-KDD est une version améliorée du dataset KDD'99, contenant 125 973 échantillons pour l'entraînement et 22 544 pour le test. Il comprend 41 attributs par échantillon et couvre quatre types d'attaques : DoS (déni de service : surcharge des ressources), Probe (sondage : collecte d'informations), R2L (accès non autorisé : exploitation de vulnérabilités), et U2R (escalade de privilèges : accès non autorisé à des privilèges élevés).
CICIDS2017 [25]	CIC-IDS2017 est un dataset réaliste de trafic réseau, généré à partir de scénarios simulant des activités normales et malveillantes sur 5 jours. Il contient environ 2,8 millions d'échantillons, chacun décrit par 80 attributs. Il couvre un large éventail d'attaques modernes, notamment DoS (déni de service : surcharge des ressources), Brute-force (force brute : tentative d'accès par essais répétés), Botnet (réseau de bots : contrôle malveillant de machines), Web attacks (attaques web : exploitation de vulnérabilités web), DDoS (déni de service distribué : attaque massive coordonnée), Infiltration (infiltration : accès furtif au système), et Heartbleed (exploitation d'une faille SSL).
KDD Cup 99 [26]	Il contient environ 4,9 millions d'échantillons et 41 attributs. Les attaques sont classées en 4 catégories principales : DoS (déni de service : surcharge des ressources), Probe (sondage : collecte d'informations), R2L (accès non autorisé : exploitation de vulnérabilités), et U2R (escalade de privilèges : accès non autorisé à des privilèges élevés).
UNSW-NB15 [27]	Dataset contenant 2,5 millions d'échantillons et 49 attributs, couvrant des attaques modernes et des activités réseau normales. Il inclut des attaques comme DoS (déni de service : surcharge des ressources), Exploitation (exploitation : utilisation de failles logicielles), Brute-force (force brute : tentative d'accès par essais répétés), Shellcode (code malveillant : exécution de commandes non autorisées), etc.
CSE-CIC-IDS2018 [28]	Il contient 2,83 millions d'échantillons et 80 attributs, couvrant des attaques telles que DDoS (déni de service distribué : attaque massive coordonnée), Brute-force (force brute : tentative d'accès par essais répétés), Botnet (réseau de bots : contrôle malveillant de machines), Web attack (attaques web : exploitation de vulnérabilités web), et plus.

TABLE 1.4 – Jeux de données couramment utilisés pour la détection d'intrusion

Limites des jeux de données utilisés

Le tableau 1.4 présente les jeux de données les plus couramment utilisés. Cependant, la majorité des jeux de données utilisés pour la détection d'intrusion présentent de nombreuses limitations susceptibles de compromettre l'efficacité des modèles d'apprentissage automatique. Une étude récente menée par [29] souligne que la plupart des ensembles de données disponibles dans ce domaine sont construits à partir de captures réalisées dans des environnements simulés ou en laboratoire. Même les jeux de données populaires, tels que *CICIDS2017*, sont basés sur du trafic artificiellement généré, ce qui limite considérablement leur réalisme. En conséquence, les modèles entraînés sur ces jeux risquent de ne pas se généraliser efficacement aux scénarios d'attaque du monde réel.

Un autre problème récurrent, également mis en lumière par [29], est le fort déséquilibre entre le trafic normal et le trafic malveillant. Le trafic légitime y est largement majoritaire, tandis que les attaques y sont rares. Cette asymétrie induit un biais dans l'apprentissage, rendant les modèles performants sur la classe majoritaire mais inefficaces pour détecter les intrusions peu fréquentes. Cela conduit souvent à un surapprentissage sur les motifs du trafic normal, au détriment de la capacité du modèle à repérer les attaques.

Pour pallier ces limitations, les auteurs de [29] recommandent une sélection rigoureuse des jeux de données, un prétraitement approprié, ainsi que des protocoles d'évaluation fidèles à des conditions réalistes.

1.3.4 Sélection des caracteristique

Définition

La sélection de caractéristiques (*feature selection*) est définie comme un processus visant à réduire la dimensionnalité d'un ensemble de données en extrayant uniquement les caractéristiques les plus informatives et pertinentes pour la prédiction d'une intrusion, tout en éliminant les caractéristiques non informatives, redondantes ou bruitées [30]. Ce processus améliore l'efficacité des systèmes de détection d'intrusion (IDS) en réduisant la complexité des modèles d'apprentissage automatique et en augmentant la précision de la détection des anomalies.

Exemple : Considérant le dataset CICIDS2017 avec 80 caractéristiques de flux réseau, le tableau suivant liste certaines caractéristiques, suivies des 70 autres :

Caractéristiques du dataset CICIDS2017									
Flow Duration	Total Fwd Packets	Avg Packet Size	PSH Flag Count	Flow Bytes/s	Flow Packets/s	Packet Len Var	Active Mean	Time stamp	+70 autres

TABLE 1.5 – Caractéristiques du dataset CICIDS2017

Après application d'un algorithme de sélection de caractéristiques, comme un

arbre de décision, seules les plus pertinentes pour détecter les attaques DoS (Par exemple) sont retenues, comme illustré ci-dessous :

Caractéristique	**Pertinence pour la détection**
Avg Packet Size	Taille réduite (< 50 octets) pour trafic automatisé
Flow Bytes/s	Débit élevé caractéristique des attaques DoS
Flow Packets/s	Taux élevé de paquets/s indique un flux malveillant
Packet Len Var	Faible variance pour paquets uniformes DoS

TABLE 1.6 – Caractéristiques retenues pour la détection d'attaques DoS

La sélection de caractéristiques élimine les données moins pertinentes, comme Timestamp, pour optimiser un classificateur, améliorant ainsi l'efficacité et la précision.

Objectifs de la sélection de caractéristiques

Dans le cadre de la détection d'intrusion, la sélection de caractéristiques vise à :

1. Réduire la dimensionnalité pour éviter le surajustement et accélérer l'entraînement.
2. Améliorer l'efficacité des calculs pour une analyse en temps réel.
3. Augmenter la précision des modèles de détection.
4. Simplifier les modèles pour une meilleure interprétation.
5. Identifier les attributs clés liés aux intrusions [30].

Méthodes de sélection de caractéristiques

Toujours selon [30], les méthodes de sélection de caractéristiques sont les suivantes :

- **Filtres univariés** : Évaluent chaque caractéristique indépendamment avec des critères comme le test du χ^2 (*Chi-squared test*), mesurant l'association avec la classe cible (par exemple, paquets réseau), et F-ANOVA (*Analysis of Variance*, statistique de Fisher pour comparer les variances), testant les différences de moyennes entre classes (par exemple, trafic normal vs intrusion). Avantages : rapidité, scalabilité. Limites : ignorent les interactions, performance prédictive limitée (χ^2), sensibilité à la normalité (F-ANOVA).
- **Filtres multivariés** : Considèrent les dépendances entre caractéristiques via ReliefF, distinguant les échantillons similaires de classes différentes, et PCA (*Principal Component Analysis*), transformant les caractéristiques en composantes orthogonales pour maximiser la variance. Limites : redondances non supprimées (ReliefF), perte d'interprétabilité (PCA).

- **Enveloppeurs** : Optimisent via une recherche heuristique, comme RFE (*Recursive Feature Elimination*), éliminant les caractéristiques les moins importantes avec SVM (*Support Vector Machine*) pour la détection d'intrusion. Limites : coût élevé, risque de surajustement, dépendance au classificateur.
- **Intégrées** : Intègrent la sélection dans l'entraînement, utilisant les forêts aléatoires (*Random Forests*, importance via *Mean Decrease Impurity*) ou LASSO (*Least Absolute Shrinkage and Selection Operator*, pénalisation des coefficients). Limites : faible détection des interactions (forêts aléatoires), dépendance au classificateur (LASSO).

1.3.5 Les métriques de performance

Les métriques de performance sont utilisées pour évaluer les performances d'un modèle sur un dataset donné. Elles fournissent des moyens quantitatifs pour tester l'efficacité d'un modèle. Elles permettent également d'identifier les domaines à améliorer et à optimiser. [31]

Explication des Métriques

Les termes utilisés dans les équations sont définis dans le tableau 1.7.

Terme	Définition
Vrais Positifs (TP pour : *True Positives*)	Instances positives correctement prédites
Vrais Négatifs (TN Pour : *True Negatives*)	Instances négatives correctement prédites
Faux Positifs (FP Pour : False Positives)	Instances négatives incorrectement prédites comme positives
Faux Négatifs (FN Pour : False Negatives)	Instances positives incorrectement prédites comme négatives
Taux de Vrais Positifs (TPR Pour : *True Positive Rate*)	Proportion de positifs réels correctement identifiés, (équivalent au Rappel)

TABLE 1.7 – Définitions des Termes de la Matrice de Confusion

1. **Exactitude** : Proportion d'instances correctement classées parmi toutes les instances.
$$\text{Exactitude} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.1}$$
2. **Précision** : Proportion de positifs prédits qui sont réellement positifs.
$$\text{Précision} = \frac{TP}{TP + FP} \tag{1.2}$$
3. **Rappel** : Proportion de positifs réels correctement identifiés.
$$\text{Rappel} = \frac{TP}{TP + FN} \tag{1.3}$$

4. **F1-Score** : Moyenne harmonique de la précision et du rappel, équilibrant les deux.

$$\text{F1-Score} = 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}} \tag{1.4}$$

5. **FPR** : Proportion de négatifs incorrectement classés comme positifs.

$$\text{FPR} = \frac{FP}{FP + TN} \tag{1.5}$$

6. **AUC(Pour : *Area Under the Curve*)** : Probabilité qu'une instance positive soit classée plus haut qu'une instance négative.

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR})\, d\text{FPR} \tag{1.6}$$

La figure 1.3 illustre la matrice de confusion obtenue lors de l'évaluation d'un modèle. Elle permet de visualiser la répartition des prédictions correctes (TP et TN) et incorrectes (FP et FN), facilitant ainsi l'interprétation des performances du modèle.

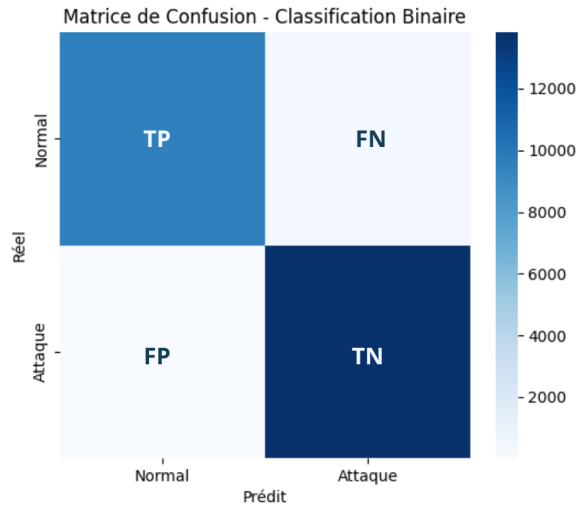


FIGURE 1.3 – Matrice de confusion pour la classification binaire

1.4 Fondements opérationnels de la détection d'intrusion

1.4.1 Collecte de données

La collecte de données constitue la première étape d'un système de détection d'intrusion. Les données peuvent être recueillies au niveau de l'hôte (journaux système, appels système) ou au niveau du réseau (paquets capturés). En pratique, on

distingue ainsi les IDS basés hôte (HIDS) et réseau (NIDS) : un IDS hôte collecte des données reflétant fidèlement l'activité locale du système, tandis qu'un IDS réseau se base sur l'analyse du trafic circulant sur le réseau [32]. Ces données brutes (flux réseau, enregistrements système, etc.) sont ensuite utilisées pour extraire des caractéristiques.

1.4.2 Définition d'une caractéristique

Une *caractéristique* (ou *attribut*) est une propriété mesurable extraite des données, utilisée pour différencier comportements normaux et anomalies. Dans le contexte des IDS, cela peut être par exemple le nombre de paquets par unité de temps, la taille moyenne des connexions, ou des métriques similaires dérivées du trafic réseau ou des journaux d'activité. Par définition générale, une caractéristique est un attribut ou une variable mesurable *d'un objet* utile à la classification [33].

Exemple : Considérons un serveur web interne dans une entreprise (adresse IP : `192.168.1.10`), surveillé par un système de détection d'intrusion réseau (NIDS). Une caractéristique clé extraite du trafic réseau est la *taille moyenne des connexions*, mesurée en kilo-octets (Ko) par connexion. Cette caractéristique est obtenue en utilisant `tcpdump`, un outil de capture de paquets réseau qui enregistre le trafic transitant par une interface réseau [34]. Par exemple, `tcpdump` peut être configuré pour capturer uniquement les paquets TCP dirigés vers le port `80` du serveur, correspondant aux requêtes HTTP. Une commande typique serait :

```
tcpdump -i eth0 host 192.168.1.10 and port 80 -w capture.pcap
```

Cette commande capture les paquets réseau à destination du serveur (`192.168.1.10`) sur le port `80` et les enregistre dans un fichier `capture.pcap`. Les données capturées sont ensuite analysées pour calculer la taille moyenne des connexions, en sommant la taille des données transférées par connexion et en divisant par le nombre total de connexions sur une période donnée (par exemple, une heure). Le tableau suivant décrit cette caractéristique :

Caractéristique	**Description**
Taille moyenne des connexions	Taille moyenne des données transférées par connexion TCP (en Ko), extraite à partir des paquets capturés par `tcpdump` sur une période donnée.

TABLE 1.8 – Caractéristique mesurable pour la détection d'anomalies

En conditions normales, les captures tcpdump indiquent un transfert moyen de 150 Ko par connexion HTTP légitime. Une connexion anormale de 10 Mo, détectée par le NIDS, peut signaler une exfiltration de données. Le NIDS déclenche alors une alerte. La caractéristique « taille moyenne des connexions », extraite via tcpdump, permet au système de comparer les valeurs observées à une base de référence pour identifier les anomalies et protéger le réseau.

1.4.3 Définition d'une attaque

Une *attaque* est habituellement définie comme toute action malveillante visant à compromettre un système.[35]. Autrement dit, une attaque a pour but de violer la confidentialité, l'intégrité ou la disponibilité des données ou des services d'un système.

Un exemple est **le débordement de tampon** (En anglais : *BUFFER OVERFLOW*), où un attaquant envoie des données excessives pour écraser la mémoire. Voici un code en C simplifié de cette attaque :

```
#include <stdio.h>
#include <string.h>
void vulnerable_function(char *input) {
    char buffer[10];
    strcpy(buffer, input); // Pas de vérification de la taille
}
int main() {
    char attack[] = "AAAAAAAAAAAAAAAAAAAA\x90\x90\x90\x90";
    vulnerable_function(attack);
    return 0;
}
```

Dans cet exemple, une entrée trop longue (*attack*) surcharge *buffer*, permettant à un attaquant d'exécuter du code malveillant. Un HIDS peut détecter cette anomalie en surveillant les modifications non autorisées de la mémoire.

1.5 Conclusion

En conclusion, ce chapitre a exploré les concepts et méthodes fondamentaux des systèmes de détection d'intrusion (IDS) et leur intégration avec l'intelligence artificielle pour renforcer la cybersécurité. Nous avons détaillé les différentes classifications des IDS, leurs avantages, ainsi que leurs limites, tout en mettant en lumière le rôle crucial de l'IA dans l'amélioration de la détection des menaces. Les concepts clés de l'IA, les techniques de prétraitement des données, les jeux de données utilisés, et les métriques de performance ont été examinés pour évaluer l'efficacité des modèles de détection.

De plus, nous avons discuté de l'importance de la collecte de données, qui est un élément essentiel pour comprendre le fonctionnement et l'efficacité des systèmes de détection d'intrusion.

Ce chapitre vise à offrir une compréhension approfondie des méthodes et des technologies utilisées pour protéger les systèmes informatiques contre les intrusions. Le chapitre suivant sera consacré à l'état de l'art, où nous examinerons les avancées récentes et les tendances actuelles dans le domaine de la détection d'intrusion et de l'application de l'IA en cybersécurité.

Chapitre 2
État de l'Art des Systèmes de Détection d'Intrusion

2.1 Introduction

Ce chapitre explore l'état de l'art des systèmes de détection d'intrusions basés sur l'apprentissage automatique et l'apprentissage profond. Nous examinons les différentes approches proposées dans la littérature récente, en mettant l'accent sur les techniques utilisées, les jeux de données employés et les performances obtenues. Nous discutons également des limites et des défis associés à ces approches, tels que la dépendance à la sélection de caractéristiques, le déséquilibre des classes, la généralisation aux attaques rares et la complexité de calcul.

Enfin, nous proposons une approche hybride combinant un réseau de neurones convolutifs (CNN) et le Swin Transformer pour surmonter certaines de ces limites. Ce chapitre vise à fournir une compréhension approfondie des avancées récentes dans le domaine et à ouvrir la voie à des recherches futures pour améliorer la détection d'intrusions dans les réseaux modernes.

2.2 Revue de la littérature

2.2.1 Intrusion detection model using machine learning algorithms on NSL-KDD dataset. (2024)

Alshamy et al. [36] proposent une méthodologie pour la détection d'intrusions à l'aide du modèle *IDS-RF*(Pour : *Intrusion Detection System-Random Forest*) sur le jeu de données *NSL-KDD*. Le processus commence par l'importation du jeu de données, suivie d'un prétraitement des données, qui inclut la transformation des caractéristiques catégoriques en numériques par un encodage *one-hot* et la standardisation des données avec *StandardScaler* pour uniformiser les échelles des caractéristiques. Pour remédier au déséquilibre des classes, la technique *SMOTE* est utilisée pour générer des échantillons synthétiques pour les classes minoritaires. Un classificateur *Random Forest (RF)* est ensuite entraîné sur les données prétraitées pour effectuer une classification binaire et multiclasse des attaques réseau (*Normal*, *DoS*, *Probe*, *R2L*, *U2R*). Le modèle est évalué sur un sous-ensemble de test du jeu de données *NSL-KDD*.

Cette approche repose sur le jeu de données *NSL-KDD*, qui est considéré comme obsolète et ne reflète pas les attaques réseau modernes. De plus, la classe *U2R* souffre d'un faible nombre d'instances, ce qui peut entraîner une sous-représentation et une précision de classification réduite pour cette catégorie. L'utilisation de *SMOTE* peut également introduire un risque de surajustement si les échantillons synthétiques ne

capturent pas pleinement la variabilité réelle des données.

2.2.2 Intrusion Detection System with Customized Machine Learning Techniques for NSL-KDD Dataset. (2023)

Zakariah et al. [37] proposent une méthodologie pour un système de détection d'intrusions (IDS) basé sur un réseau de neurones artificiels (*ANN*) utilisant le jeu de données *NSL-KDD*. La méthode débute par un prétraitement des données, incluant la suppression des caractéristiques inutiles, la discrétisation des valeurs continues et l'élimination des entrées redondantes. Une analyse en composantes principales (*PCA*, En anglais : *Principal Component Analysis*) réduit la dimensionnalité tout en préservant la variabilité des données, facilitant le traitement par l'IDS. L'*ANN*, implémentée avec *Keras*, utilise des couches denses, des fonctions d'activation *ReLU*(Pour : *Rectified Linear Unit*) pour les couches cachées et *sigmoid* pour la couche de sortie, avec une régularisation par *dropout* pour éviter l'*overfitting*. Les données sont divisées en ensembles d'entraînement, de validation et de test, avec un réglage des hyperparamètres via *Keras Tuner*. Les performances sont évaluées à l'aide de métriques telles que l'exactitude, la précision, le rappel et le score *F1*.

Le jeu de données *NSL-KDD* peut ne pas refléter la complexité du trafic réseau réel, limitant la généralisation. La *PCA* risque de manquer des relations non linéaires, ce qui pourrait réduire la précision de détection. La complexité de l'*ANN* varie selon la configuration, et le déséquilibre des données peut biaiser les prédictions. La scalabilité et les performances peuvent être contraintes par le matériel, et l'adaptabilité à de nouveaux types d'attaques reste incertaine.

2.2.3 A hybrid CNN-LSTM approach for intelligent cyber intrusion detection system. (2025)

Bamber et al. [38] proposent une méthodologie pour la détection d'intrusions basée sur un modèle hybride *CNN-LSTM* utilisant le jeu de données *NSL-KDD*. Le processus débute par l'importation des données, suivie d'un prétraitement pour préparer les caractéristiques et les étiquettes pour le modèle *CNN-LSTM*. Une sélection de caractéristiques est effectuée en deux étapes : une méthode univariée utilisant un test *ANOVA F-test* pour évaluer la pertinence des caractéristiques, suivie d'une élimination récursive des caractéristiques (*RFE*) avec un classificateur basé sur un *Decision Tree* pour identifier le sous-ensemble optimal de caractéristiques. Le modèle *CNN-LSTM* est ensuite entraîné sur les données prétraitées pour classer les activités réseau comme normales ou anormales (*Normal*, *DoS*, *Probe*, *R2L*, *U2R*). Les performances sont évaluées à l'aide de métriques telles que l'exactitude, la précision, le rappel et le score *F1* et la courbe *ROC*(Pour : *Receiver Operating Characteristic*), avec des tests sur des ensembles de validation du *NSL-KDD*.

Cette approche repose sur le jeu de données *NSL-KDD*, qui est obsolète et ne représente pas les attaques réseau contemporaines, limitant ainsi la généralisation du modèle à des scénarios réels. La complexité de calcul du modèle *CNN-LSTM*, due à la combinaison de couches convolutionnelles et récurrentes, peut poser des défis pour une utilisation en temps réel. De plus, bien que la sélection de caractéristiques

par *RFE* améliore l'efficacité, elle peut exclure des caractéristiques potentiellement pertinentes, affectant la robustesse du modèle dans certains cas.

2.2.4 Deep learning algorithms for intrusion detection systems in internet of things using CIC-IDS 2017 dataset. (2023)

Jose et al. [39] proposent une méthodologie pour les systèmes de détection d'intrusions dans l'Internet des objets (IoT) en utilisant des algorithmes d'apprentissage profond, testés sur le jeu de données *CIC-IDS2017*. Le processus commence par le prétraitement des données, où huit fichiers CSV sont fusionnés en un seul ensemble, suivi d'un nettoyage pour éliminer les caractères spéciaux, les valeurs infinies (1 509 dans *flowbytes/s*, 2 867 dans *flowpackets/s*) et les valeurs nulles, remplacées par des zéros. Une analyse exploratoire des données (*EDA* Pour : *Exploratory Data Analysis*) est effectuée, incluant une réduction de dimensionnalité via l'analyse en composantes principales (*PCA*), réduisant les caractéristiques à 71, suivies d'un encodage des étiquettes et d'une normalisation. Les données sont divisées en ensembles d'entraînement (70%) et de test (30%).

Trois modèles d'apprentissage profond sont implémentés : un réseau neuronal profond (*DNN* Pour : *Deep Neural Network*), un réseau à mémoire à court et long terme (*LSTM*) et un réseau convolutionnel (*CNN*). Le *DNN* comprend une couche d'entrée de 250 neurones, trois couches cachées (32, 72, 32 neurones) avec activation *ReLU*, et une couche de sortie avec activation *SoftMax*, utilisant l'entropie croisée catégorique comme fonction de perte et l'optimiseur *Adam*. Le *LSTM* inclut quatre couches cachées (64, 64, 128, 128 neurones) avec *ReLU* et *SoftMax*, adapté aux données temporelles via des portes d'oubli, d'entrée et de sortie. Le *CNN* possède trois couches cachées (120, 60, 30 neurones) avec *ReLU*, des couches de *MaxPooling* (taille 2), et une couche de sortie avec 15 neurones, utilisant l'entropie croisée catégorique sparse. Les performances sont évaluées via l'exactitude, la précision, le rappel et le score *F1*.

L'approche présente plusieurs limites. Le jeu de données utilisé, bien que plus récent que d'autres, ne couvre pas de manière exhaustive les menaces spécifiques aux environnements IoT actuels, ce qui compromet la généralisation des résultats. La complexité de calcul des modèles, en particulier les réseaux *CNN* et *LSTM*, soulève également des problèmes de faisabilité sur des dispositifs IoT (Pour :*Internet of Things*) aux ressources limitées. Par ailleurs, la réduction de dimensionnalité par *PCA* peut entraîner une perte d'information utile à la classification, et certaines décisions de prétraitement, telles que le remplacement des valeurs nulles par des zéros, peuvent introduire des biais ou fausser l'interprétation des données.

2.2.5 Système de détection d'intrusions basé sur les *Deep Belief Networks* (2024)

Belarbi et al. [40] présentent un système de détection d'intrusions réseau (*NIDS*) fondé sur les *Deep Belief Networks* (*DBNs*), en utilisant le jeu de données *CICIDS2017*. Le traitement des données comprend une réduction de la dimensionnalité par analyse en composantes principales (*PCA*), visant à simplifier l'espace des caractéris-

tiques tout en maintenant un niveau d'information acceptable. Les données sont réparties en ensembles d'entraînement, de validation et de test à l'aide d'une séparation stratifiée, afin de mieux gérer le déséquilibre entre classes. Plusieurs techniques de rééquilibrage sont mises en œuvre : sous-échantillonnage, sur-échantillonnage par *SMOTE*, et pondération des classes ou des échantillons.

L'entraînement du modèle *DBN* suit un processus en deux phases : un pré-entraînement non supervisé couche par couche à l'aide de *Restricted Boltzmann Machines* (*RBMs*), suivi d'un ajustement supervisé par rétropropagation. Une évaluation comparative est menée avec un modèle *Multi-Layer Perceptron* (*MLP*) ainsi qu'avec d'autres approches existantes.

Le recours à un échantillonneur centralisé constitue une limitation dans des contextes impliquant de larges volumes de données, des exigences de confidentialité ou des infrastructures réseau distribuées. En outre, le déséquilibre marqué des classes dans les jeux de données comme *CICIDS2017* complique la détection des attaques rares. L'utilisation de la *PCA*, bien qu'efficace pour la réduction de dimensionnalité, peut entraîner une perte d'information susceptible d'affecter les performances du modèle sur les cas moins représentés.

2.2.6 Efficient Deep Neural Network for Intrusion Detection Using CIC-IDS-2017 Dataset. (2024)

L'auteur Bandarupalli [41] propose un système de détection d'intrusions réseau (*NIDS*) basé sur un *Deep Neural Network* (*DNN*) de type anomalie, utilisant le jeu de données *CIC-IDS-2017*. Le prétraitement des données comprend une sélection de caractéristiques via *ANOVA* pour réduire la dimensionnalité à 36 caractéristiques, éliminant les éléments non essentiels comme les adresses IP et les horodatages. La normalisation des données est effectuée avec la transformation *Yeo-Johnson* pour gérer les valeurs positives et négatives et réduire les biais. Pour traiter le déséquilibre des classes, des techniques de sur-échantillonnage (*SMOTE*) et de sous-échantillonnage aléatoire (*RUS*) sont appliquées. Le modèle *DNN* utilise quatre couches cachées avec une fonction d'activation *ReLU*, une fonction *softmax* pour la classification multi-classe, et des couches de *dropout* pour éviter le surajustement. Une comparaison est réalisée avec d'autres architectures comme *HAST-IDS*(Pour : *Hybrid Attention-based Spatio-Temporal Intrusion Detection System*), *FESVDF*(Pour : *Feature Selection using Singular Value Decomposition*), et *SHIA*(Pour : *Security Hybrid Intelligent Approach*).

La méthode présente des limites en termes de précision pour certaines classes d'attaques, comme *DoS Hulk* et *Web Attack Brute Force*, où des faux négatifs sont observés. Le modèle a également du mal à distinguer les paquets bénins des attaques, entraînant des faux positifs, un problème courant dans les *NIDS* basés sur les anomalies. De plus, bien que plus léger que d'autres architectures, sa performance est inférieure à des modèles plus complexes comme *SGM-CNN* (Pour : Stochastic Gradient Method with Convolutional Neural Network) ou *AI-SIEM*(Pour : Artificial intelligence-Security Information and Event Management), qui utilisent davantage de ressources de calcul.

2.2.7 A One-Dimensional Convolutional Neural Network (1D-CNN) Based Deep Learning System for Network Intrusion Detection. (2022)

Les auteurs Qazi et al. [42] proposent un système de détection d'intrusions réseau (*NIDS*) basé sur un réseau neuronal convolutif unidimensionnel (*1D-CNN* Pour : *One-Dimensional Convolutional Neural Network*), en utilisant le jeu de données *CICIDS2017*. Le système cible quatre types d'attaques : *DoS Hulk*, *DoS GoldenEye*, *DDoS* (actives) et *PortScan* (passive). Le prétraitement des données inclut la fusion de huit fichiers en un seul, la suppression de 135 000 valeurs manquantes ou infinies, et l'exclusion de caractéristiques comme les adresses IP et les horodatages, réduisant les 83 caractéristiques initiales à 79. Pour gérer le déséquilibre des classes, un sous-échantillonnage aléatoire est appliqué, limitant les instances à 50 000 pour *BENIGN* et entre 10 000 et 25 000 pour les attaques. L'architecture *1D-CNN* comprend cinq couches convolutives avec 32 à 16 filtres, une couche de *dropout* (taux de 0,5), une couche de *max-pooling*, et cinq couches denses avec *ReLU* et *SoftMax* pour la classification multiclasse.

Les limites incluent une dépendance à un sous-ensemble spécifique du jeu de données, ce qui peut affecter la généralisation à d'autres types d'attaques. Le sous-échantillonnage réduit la diversité des données, potentiellement au détriment de la détection d'attaques rares. Comparé à d'autres approches comme *LSTM* ou *DNN*, le modèle *1D-CNN* est moins coûteux en calcul, mais des analyses futures avec des techniques comme *PCA* ou *ICA* sont envisagées pour réduire davantage les caractéristiques.

2.2.8 Optimizing neural networks using spider monkey optimization algorithm for intrusion detection system. (2024)

Kumari et al. [43] ont développé une méthodologie visant à optimiser les réseaux neuronaux artificiels (ANN) en utilisant l'algorithme d'optimisation inspiré de la nature, dénommé optimisation des singes araignées (SMO Pour : *Spider Monkey Optimization*), pour les systèmes de détection d'intrusion. La méthode, appelée SMO-ANN, débute par un prétraitement des données, comprenant la suppression des valeurs manquantes, la normalisation des échantillons et l'encodage des étiquettes. Les couches de l'ANN sont ensuite optimisées par SMO, qui s'inspire du comportement de recherche alimentaire des singes araignées, en ajustant les positions des solutions à travers des phases locale et globale pour améliorer les performances du modèle. Les données sont partitionnées en ensembles d'entraînement et de test afin d'évaluer la capacité du modèle à classifier le trafic réseau comme bénin ou malveillant. Toutefois, cette approche présente des limites, notamment une sensibilité potentielle au déséquilibre des classes, ce qui peut compromettre la détection des attaques rares, ainsi que des préoccupations liées à la confidentialité des données, nécessitant des mesures supplémentaires pour une utilisation dans des secteurs sensibles.

2.2.9 CNN-BiLSTM : A Hybrid Deep Learning Approach for Network Intrusion Detection System in Software-Defined Networking With Hybrid Feature Selection. (2023)

Ben Said et al. [44] ont proposé une méthodologie pour améliorer la détection d'intrusions dans les réseaux définis par logiciel (SDN Pour : *Software-Defined Networking*) en utilisant un modèle hybride combinant des réseaux neuronaux convolutionnels (CNN) et une mémoire à court et long terme bidirectionnelle (BiLSTM Pour : *Bidirectional Long Short-Term Memory*). La méthode, nommée CNN-BiLSTM, débute par un prétraitement des données, incluant la suppression des valeurs manquantes, la normalisation des échantillons et l'encodage des étiquettes. Une sélection hybride des échantillons est réalisée à l'aide d'un classificateur de forêt aléatoire et d'une élimination récursive des échantillons, réduisant leur nombre pour optimiser les performances. Le modèle CNN extrait les caractéristiques spatiales, tandis que BiLSTM capture les caractéristiques temporelles, améliorant ainsi la classification binaire et multiclasse. Les données sont divisées en ensembles d'entraînement et de test pour évaluer la capacité du modèle à détecter des attaques telles que DDoS, U2R ou R2L, en utilisant les ensembles de données InSDN (Pour : *Intrusion Detection in Software-Defined Networking*), NSL-KDD et UNSW-NB15. Cependant, cette approche présente des limites, notamment un temps d'entraînement prolongé et un risque potentiel de surajustement lié au suréchantillonnage aléatoire.

2.2.10 Stacked Ensemble Deep Learning for Robust Intrusion Detection in IoT Networks. (2025)

Amara et al. [45] ont proposé une méthodologie pour renforcer la détection d'intrusions dans les réseaux d'Internet des objets (IoT) en utilisant un modèle d'ensemble empilé combinant des réseaux neuronaux convolutionnels (CNN), des réseaux convolutionnels temporels (TCN Pour : *Temporal Convolutional Network*) et une mémoire à court et long terme (LSTM). La méthode débute par un prétraitement des données, incluant la suppression des valeurs manquantes, des doublons, des colonnes à variance nulle, et la normalisation des échantillons. Pour traiter le déséquilibre des classes, trois techniques de rééchantillonnage sont appliquées : le suréchantillonnage, le sous-échantillonnage et une approche intermédiaire (Meet-in-the-Middle). Le modèle CNN capture les motifs spatiaux, le TCN modélise les dépendances temporelles à long terme, et le LSTM identifie les motifs séquentiels. Les prédictions des modèles de base sont combinées via une régression logistique, formant un méta-modèle qui optimise la classification binaire entre trafic bénin et malveillant. Les données sont divisées en ensembles d'entraînement (80%) et de test (20%) pour évaluer la détection d'attaques sur le jeu de données CICIDS2017. Cependant, cette approche présente plusieurs limites, notamment une complexité accrue et un temps d'entraînement prolongé.

2.2.11 DS-kNN : An Intrusion Detection System Based on a Distance Sum-Based K-Nearest Neighbors. (2021)

Taguelmimt et al. [46] ont proposé une méthodologie pour améliorer la détection d'intrusions dans les réseaux en utilisant une version optimisée du classificateur k-plus proches voisins, nommée DS-kNN. La méthode commence par une sélection des caractéristiques les plus pertinentes, réduisant les 41 caractéristiques du jeu de données KDD Cup 99, sur la base du gain d'information. Pour chaque échantillon de données à classer, DS-kNN (Pour : *Distance Sum-based k-Nearest Neighbors*) identifie les k voisins les plus proches dans chaque classe possible (normal, DoS, Probe, R2L, U2R) et calcule la somme des distances euclidiennes entre l'échantillon et ces voisins. L'échantillon est ensuite assigné à la classe ayant la plus petite somme. Les données sont divisées en ensembles d'entraînement (494 022 échantillons) et de test (311 029 échantillons) pour évaluer la détection des attaques. L'approche présente des limites pour détecter les classes R2L et U2R en raison de leur faible représentation dans l'ensemble d'entraînement.

2.2.12 IDS-INT : Intrusion detection system using transformer-based transfer learning for imbalanced network traffic. (2024)

Ullah et al. [47] proposent une méthode pour la détection d'intrusions réseau, appelée IDS-INT, basée sur l'apprentissage par transfert utilisant des transformers pour traiter le trafic réseau déséquilibré. La méthodologie commence par l'utilisation d'un modèle transformer (BERT large Pour : *Bidirectional Encoder Representations from Transformers Large*) pour extraire des relations complexes entre les caractéristiques des données réseau, telles que les références d'attaques et les informations d'hôtes. Ensuite, la méthode SMOTE est appliquée pour équilibrer les données en augmentant les instances d'attaques minoritaires. Un modèle CNN est développé pour extraire des caractéristiques profondes à partir des données équilibrées, suivi d'un modèle hybride CNN-LSTM pour apprendre et détecter différents types d'attaques. Une approche d'intelligence artificielle explicable (XAI, En anglais : *Explainable Artificial Intelligence*) est intégrée pour interpréter les contributions des caractéristiques via des outils comme LIME (En anglais : *Local Interpretable Model-agnostic Explanations*) et SHAP (En anglais : *SHapley Additive exPlanations*). Les inconvénients incluent une dépendance à la qualité des caractéristiques sémantiques, une efficacité limitée face à des données très déséquilibrées, et une complexité de calcul élevée due à l'utilisation de modèles profonds et de techniques d'interprétation.

2.2.13 HiViT-IDS : An Efficient Network Intrusion Detection Method Based on Vision Transformer. (2025)

Zhou et al. [48] proposent une méthode de détection d'intrusions réseau, appelée HiViT-IDS (Pour : *Hybrid Vision Transformer for Intrusion Detection System*), basée sur le Vision Transformer (ViT) pour traiter le trafic réseau dans les environnements IoT. La méthodologie commence par le prétraitement des données, éliminant les caractéristiques inutiles et codant les caractéristiques textuelles via Label Encoding.

Ensuite, les données unidimensionnelles sont transformées en images RGB à l'aide de la normalisation quantile, chaque image étant formée à partir de blocs de données consécutifs (par exemple, 126 lignes pour ToN-IoT, 285 pour Edge-IIoTset). Ces images sont redimensionnées à 224x224x3 et introduites dans un modèle ViT, qui utilise un mécanisme d'auto-attention multi-têtes et un encodeur Transformer pour la classification. Les ensembles de données utilisées sont : ToN-IoT et Edge-IIoTset. Les inconvénients incluent l'absence de tests sur des échantillons adversariaux et une dépendance à la densité d'information des ensembles de données.

2.2.14 FlowTransformer : A transformer framework for flow-based network intrusion detection systems. (2024)

Manocchio et al. [49] proposent le cadre FlowTransformer, une approche novatrice pour implémenter des systèmes de détection d'intrusions réseau (NIDS) basés sur des transformateurs, en exploitant les flux réseau (NetFlow). La méthodologie repose sur trois composants principaux : un encodage d'entrée transformant les flux réseau en vecteurs de longueur fixe, des blocs de transformateurs (encodeurs ou décodeurs) capturant les relations complexes entre les flux, et une tête de classification convertissant la sortie séquentielle en une prédiction. FlowTransformer permet l'évaluation systématique de différentes configurations, incluant l'encodage d'entrée, le modèle de transformateur, et la tête de classification, sur des ensembles de données variés comme NSL-KDD, UNSW-NB15, et ToN-IoT. Parmi les limites, on note que l'entraînement peut être instable et dépend fortement du réglage du taux d'apprentissage. De plus, des aspects importants comme l'explication des décisions du modèle ou sa capacité à s'adapter aux changements dans les données ne sont pas intégrés.

2.2.15 Network Intrusion Detection via Flow-to-Image Conversion and Vision Transformer Classification. (2022)

Ho et al. [50] proposent une méthode de détection d'intrusions réseau basée sur la conversion de flux de données réseau en images RGB, suivie d'une classification par un Vision Transformer (ViT). La méthodologie comprend trois modules principaux : le prétraitement des données, la conversion de flux en images et la classification. Dans le module de prétraitement, un algorithme d'arbre de décision est utilisé pour sélectionner les 24 caractéristiques les plus pertinentes à partir de datasets comme CIC IDS2017 et UNSW-NB15, en se basant sur le gain d'information (IG). Ces caractéristiques sont ensuite extraites des fichiers CSV, et les données sont ajustées pour garantir un nombre suffisant de lignes pour former une image carrée. Le module de conversion transforme ces données en images RGB 24x24x3 via un mappage linéaire des valeurs des caractéristiques vers une plage de couleurs 24 bits, suivi d'un mécanisme de fenêtrage et de chevauchement pour standardiser la taille des images. Enfin, un classificateur ViT, pré-entraîné sur ImageNet-21K et affiné avec l'optimiseur AdamW, est utilisé pour classer les images en flux bénins ou malveillants, pour des tâches de classification binaire et multi-classes. Cependant, cette approche présente des limites significatives. Le déséquilibre des données

dans les datasets, notamment dans UNSW-NB15 où plus de 80 % des images sont bénignes, affecte gravement la précision de la classification multi-classes, en particulier pour les types d'attaques rares comme les vers ou les *shellcodes*. De plus, la sélection des caractéristiques dépend fortement de la taille des données par catégorie, ce qui peut exclure des caractéristiques pertinentes pour les attaques peu fréquentes, réduisant ainsi l'efficacité de la détection pour ces catégories. Enfin, la méthode ne prend pas en compte les corrélations temporelles entre les flux de données au-delà du fenêtrage, ce qui pourrait limiter sa capacité à détecter des attaques complexes ou évolutives.

2.3 Tableau comparatif

Le tableau 2.1 propose une analyse comparative des systèmes de détection d'intrusions réseau récents issus de la littérature. Il classe les approches selon leur jeu de données, les techniques mises en œuvre et les stratégies appliquées pour corriger le déséquilibre des classes.

Auteur(s) et Année	**Techniques utilisées**	**Jeu de données**	**Méthode de gestion du déséquilibre**	**Risque de perte d'information**
Alshamy et al. (2024) [36]	Random Forest	NSL-KDD (obsolète)	SMOTE utilisé	Possible avec SMOTE
Zakariah et al. (2023) [37]	ANN	NSL-KDD (obsolète)	Non mentionné	Possible avec PCA
Bamber et al. (2025) [38]	CNN-LSTM, RFE	NSL-KDD (obsolète)	Non mentionné	Possible avec RFE
Jose et al. (2023) [39]	CNN, LSTM, DNN	CIC-IDS2017	Non mentionné	Possible avec ANOVA
Belarbi et al. (2024) [40]	DBN	CICIDS2017	SMOTE et sous-échantillonnage	Possible avec PCA
Bandarupalli (2024) [41]	DNN	CICIDS2017	SMOTE et RUS utilisés	Possible avec ANOVA
Qazi et al. (2022) [42]	1D-CNN	CICIDS2017	Sous-échantillonnage	Possible avec réduction manuelle
Kumari et al. (2024) [43]	SMO-ANN	NSL-KDD	Non mentionné	Non mentionné
Ben Said et al. (2023) [44]	CNN-BiLSTM	InSDN, NSL-KDD, UNSW-NB15	Suréchantillonnage	Possible avec RFE
Amara et al. (2025) [45]	CNN-TCN-LSTM	CICIDS2017	Suréchantillonnage, sous-échantillonnage, Meet-in-the-Middle	Non mentionné
Taguelmimt et al. (2021) [46]	DS-kNN	KDD Cup 99	Non mentionné	Possible avec sélection de caractéristiques
Ullah et al. (2024) [47]	BERT, CNN-LSTM	CIC-IDS-2017	SMOTE utilisé	Non mentionné
Zhou et al. (2025) [48]	Vision Transformer	ToN-IoT, Edge-IIoTset	Non mentionné	Possible avec transformation en images
Manocchio et al. (2024) [49]	FlowTransformer	NSL-KDD, UNSW-NB15, ToN-IoT	Non mentionné	Non mentionné
Ho et al. (2022) [50]	Vision Transformer	CIC-IDS2017, UNSW-NB15	Sous-échantillonnage	Possible avec sélection par arbre de décision

TABLE 2.1 – Comparaison des systèmes de détection d'intrusions réseau

2.4 Discussion des limites

Les travaux récents sur les systèmes de détection d'intrusions (IDS) basés sur l'apprentissage profond ont démontré des avancées significatives, mais plusieurs limites persistent, limitant leur applicabilité dans des scénarios réels.

Premièrement, de nombreuses approches reposent sur une **sélection de caractéristiques** manuelle ou automatique, comme l'élimination récursive des caractéristiques (RFE) utilisée par Bamber et al. [38] ou l'analyse en composantes principales (PCA) par Zakariah et al. [37] et Jose et al. [39]. Ces techniques, bien qu'efficaces pour réduire la dimensionnalité, augmentent la complexité du pipeline et peuvent exclure des caractéristiques pertinentes, compromettant l'apprentissage de bout en bout et la robustesse des modèles face à des attaques complexes [38, 39].

Ensuite, le **déséquilibre des classes** est une problématique récurrente, particulièrement pour les classes rares comme U2R et R2L dans NSL-KDD ou DoS Hulk dans CIC-IDS-2017 [36, 41]. Des techniques comme SMOTE [36, 40, 47] ou le sous-échantillonnage aléatoire [42, 41] sont souvent employées pour compenser ce déséquilibre. Cependant, SMOTE peut introduire des échantillons synthétiques peu représentatifs, augmentant le risque de surajustement, tandis que le sous-échantillonnage réduit la diversité des données, affectant la généralisation [40, 42].

De plus, la **généralisation aux attaques rares** reste un défi majeur. Les modèles comme ceux de Taguelmimt et al. [46] ou Ben Said et al. [44] peinent à détecter les classes sous-représentées (par exemple, R2L et U2R dans NSL-KDD ou KDD Cup 99), malgré des performances globales satisfaisantes. Ce problème est exacerbé par l'utilisation de jeux de données obsolètes comme NSL-KDD, qui ne reflètent pas les attaques réseau modernes [36, 38, 49].

Enfin, la **complexité de calcul** des architectures profondes limite leur déploiement dans des environnements à ressources contraintes, comme les réseaux IoT. Les modèles hybrides comme CNN-TCN-LSTM [45], CNN-BiLSTM [44], ou ceux basés sur BERT [47] offrent des performances élevées mais nécessitent des ressources importantes, rendant leur utilisation en temps réel difficile, notamment pour les dispositifs IoT [39, 48].

Face à ces limites, nous proposons une approche hybride combinant un réseau de neurones convolutifs (CNN) et le Swin Transformer [51]. Cette méthode présente les avantages suivants :

- **Apprentissage de bout en bout** : En évitant la sélection manuelle de caractéristiques ou la réduction de dimensionnalité (par exemple, PCA ou RFE), le modèle apprend directement à partir des données brutes, capturant des motifs complexes sans perte d'information.
- **Gestion du déséquilibre des classes** : Une pondération adaptative des classes est utilisée pour prioriser les attaques rares sans recourir à des techniques comme SMOTE, réduisant ainsi les biais introduits par les échantillons synthétiques.
- **Transformation des données** : Les données tabulaires sont converties en images en niveaux de gris, permettant l'application efficace des architectures de vision comme le Swin Transformer, qui excelle dans la capture des dépendances globales.

- **Efficacité de calcul** : Le Swin Transformer, avec son mécanisme d'attention à fenêtre décalée, est plus économe en ressources que les Transformers classiques [51, 52], rendant le modèle adapté aux environnements IoT.

Ce choix architectural permet de capturer à la fois les motifs locaux (via CNN) et les dépendances globales (via Swin Transformer), offrant ainsi une meilleure détection même dans des conditions de données déséquilibrées. Il ouvre également la voie à des extensions futures vers des modèles d'apprentissage continu ou non supervisé, mieux adaptés à la détection d'attaques inconnues.

2.5 Conclusion

En conclusion, ce chapitre a examiné les avancées et les défis actuels des systèmes de détection d'intrusions utilisant l'apprentissage automatique et profond. Nous avons discuté des différentes techniques utilisées, des jeux de données employés, et des performances obtenues, tout en soulignant les défis associés à ces approches.

Pour surmonter certaines de ces limites, nous avons proposé une approche hybride combinant un réseau de neurones convolutifs (CNN) et le Swin Transformer.

Dans le chapitre suivant, nous introduirons la méthodologie détaillée de notre approche, en décrivant les étapes de prétraitement des données, l'architecture du modèle, et les techniques d'évaluation utilisées. Nous présenterons également les résultats expérimentaux obtenus et discuterons de leur pertinence pour la détection d'intrusions dans les réseaux modernes. Cette étude vise à ouvrir la voie à des recherches futures pour améliorer la détection d'intrusions et renforcer la sécurité des infrastructures réseau.

Chapitre 3
Approche Hybride CNN-Swin Transformer pour la Détection d'Intrusion

3.1 Introduction

La méthodologie proposée implique quatre phases principales : la préparation des données, la transformation en images, l'entraînement du modèle et l'évaluation. Cette approche est conçue pour gérer à la fois les tâches de classification binaire (normal vs attaque) et multiclasse (types d'attaques spécifiques), ce qui la rend adaptée à une large gamme de scénarios de détection d'intrusion.

Un élément clé de cette méthodologie réside dans la transformation des données tabulaires en images. Cette étape permet de tirer parti des modèles de vision par ordinateur, qui ont montré des performances remarquables en apprentissage profond. En convertissant les vecteurs de caractéristiques issus du trafic réseau en représentations visuelles, on rend possible l'utilisation de modèles tels que les CNN ou les Swin Transformers. Cette transformation facilite l'identification automatique de motifs spatiaux complexes dans les données, qui seraient difficilement capturables par des approches classiques. Elle permet également une meilleure généralisation en exploitant la structure spatiale implicite des données transformées.

Afin d'exploiter les capacités de l'apprentissage profond pour cette tâche, deux types d'architectures seront utilisés : les Réseaux de Neurones Convolutifs (CNN) et le Swin Transformer, un modèle basé sur les transformers adapté à la vision par ordinateur. Le CNN est efficace pour l'extraction de caractéristiques locales, tandis que le Swin Transformer permet de capturer des relations globales dans les images de manière hiérarchique et efficiente.

Ce chapitre est structuré comme suit : nous commencerons par une présentation succincte des modèles CNN et Swin Transformer. Ensuite, nous décrirons les différentes étapes de la méthodologie proposée, en détaillant le processus de préparation des données, la conversion de ces données en images exploitables, l'architecture du modèle combiné, le processus d'entraînement, ainsi que les méthodes d'évaluation. Enfin, un algorithme et un diagramme résumeront l'approche, suivis d'une présentation détaillée des quatre phases constituant la méthodologie.

3.2 Réseaux de Neurones Convolutifs (CNN)

Les réseaux de neurones convolutifs (CNN) sont une architecture d'apprentissage profond conçue pour traiter des données structurées spatialement, telles que

des images. Un CNN se compose généralement de couches convolutives, de couches de pooling et de couches entièrement connectées. Les couches convolutives extraient automatiquement des caractéristiques locales en appliquant des filtres aux données d'entrée, permettant une hiérarchie de représentations de plus en plus abstraites [53, 54].

Chaque couche convolutive apprend des motifs spécifiques (bords, textures, formes), et les couches de pooling réduisent la dimensionnalité des cartes de caractéristiques. Enfin, les couches entièrement connectées utilisent ces représentations pour faire des prédictions finales [55].

Les CNN sont de plus en plus utilisés dans les systèmes de détection d'intrusion (IDS), comme le montre la section 2, car ils offrent plusieurs avantages clés pour cette application :

- **Extraction automatique de caractéristiques** : Ils remplacent l'ingénierie manuelle des caractéristiques par une extraction automatisée directement à partir des données de trafic réseau [56].
- **Apprentissage des motifs spatio-temporels** : Ils capturent les relations locales entre les paquets ou les octets, aidant à identifier les motifs d'attaque [57].
- **Modélisation visuelle** : Certains travaux transforment les flux réseau en images, permettant aux CNN d'exploiter leurs forces dans l'analyse visuelle [58].

Les CNN sont également compatibles avec une exécution rapide sur les GPU (Pour : Graphics Processing Unit), facilitant leur utilisation dans des environnements en temps réel.

Les CNN émergent comme une solution efficace et puissante pour les IDS. Leur capacité à extraire automatiquement des représentations discriminatives et à détecter des motifs complexes dans le trafic réseau les rend adaptés à la détection d'intrusion moderne, en particulier dans des contextes où les menaces évoluent constamment.

3.3 Le modele Swin Transformer

Le **Swin Transformer** (*Shifted Window Transformer*) est un type d'architecture visuelle de Transformer hiérarchique introduite par Liu *et al.* [51]. Contrairement aux Vision Transformers (ViT) plats, qui appliquent une attention globale dès la première couche, Swin segmente l'image en patches locaux et applique une attention au sein de fenêtres non chevauchantes. Dans chaque bloc, ces fenêtres sont décalées ("fenêtres décalées") pour permettre la communication entre les régions adjacentes. Cela réduit considérablement la complexité de calcul tout en permettant une modélisation hiérarchique des caractéristiques visuelles.

L'architecture suit une approche pyramidale en quatre *étapes*, avec une réduction progressive de la résolution spatiale par des opérations de *fusion de patches* et une augmentation simultanée de la profondeur des caractéristiques.

Comparé aux Transformers classiques (comme ViT [52]), Swin présente plusieurs différences structurelles :

- **Attention Locale vs Globale** : Swin utilise une attention limitée aux fenêtres locales de taille fixe ($M \times M$), réduisant la complexité de calcul de $\mathcal{O}((HW)^2)$ à $\mathcal{O}(HW \cdot M^2)$.
- **Fenêtres Décalées** : Pour capturer les relations inter-fenêtres, les blocs Swin alternent entre des partitions standard et des partitions décalées de moitié.
- **Hiérarchie Multi-échelle** : Contrairement aux ViTs qui maintiennent une résolution constante, Swin construit une pyramide de résolutions, similaire aux CNN.
- **Efficacité de calcul** : Cette structure permet une meilleure évolutivité à haute résolution tout en maintenant des performances de pointe sur les tâches de vision.

Ces innovations ont permis aux Swin Transformers d'atteindre des performances de pointe en classification d'images, détection et segmentation, surpassant les CNN et les ViTs sur de nombreux benchmarks [51].

Les systèmes de détection d'intrusion (IDS) bénéficient de certaines qualités essentielles du Swin Transformer :

- **Bonne Généralisation** : L'attention hiérarchique permet de capturer à la fois les détails fins et les motifs globaux dans les données réseau, utiles pour détecter les attaques complexes [59].
- **Hiérarchie Multi-niveaux** : Un IDS doit détecter les motifs d'attaque distribués sur plusieurs paquets. La structure hiérarchique de Swin est bien adaptée à cette analyse multi-échelle.
- **Adaptation aux Données Structurées** : Bien que conçu pour la vision, les Swins peuvent être appliqués aux données réseau transformées en matrices (ou "images") représentant des paquets, des trames ou des flux, comme démontré dans plusieurs travaux récents [59].

Ainsi, la synergie entre les Swin Transformers et les CNN constitue une combinaison prometteuse, exploitant la capacité de Swin à modéliser des relations spatiales complexes et celle des CNN à extraire des motifs locaux efficaces. Cette complémentarité en fait un choix pertinent pour une analyse approfondie des données réseau.

3.4 Les etapes de la methodologie

L'approche CNN-Swin Transformer suit ces étapes clés :

1. **Préparation des Données** : Prétraiter les ensembles de données NSL-KDD et CIC-IDS-2017 en normalisant les caractéristiques, en encodant les variables catégorielles, en divisant les données et en calculant les poids des classes pour traiter le déséquilibre.
2. **Transformation en Images** : Convertir les données tabulaires en images en niveaux de gris (11x11 pour NSL-KDD, 9x9 pour CIC-IDS-2017) pour permettre le traitement basé sur la vision.

3. **Entraînement du Modèle** : Entraîner le modèle hybride CNN-Swin Transformer en utilisant les paramètres configurés, exploitant les CNN pour les caractéristiques locales et les Swin Transformers pour les dépendances globales.
4. **Évaluation** : Évaluer le modèle sur les ensembles de test, en calculant des métriques telles que la précision, la précision, le rappel, le F1-score et le taux de faux positifs pour les tâches binaires et multiclasses.

3.4.1 Préparation des Données et Transformation en Images

Les jeux de données NSL-KDD et CIC-IDS-2017 contiennent des données tabulaires avec des caractéristiques numériques et catégoriques par connexion réseau. NSL-KDD inclut 41 caractéristiques initiales, augmentées à 122 après encodage one-hot, tandis que CIC-IDS-2017 inclut 78 caractéristiques. Pour exploiter les CNN et les Swin Transformers, conçus pour les données spatiales, ces caractéristiques sont transformées en images carrées en niveaux de gris de taille 11×11 pour NSL-KDD et 9×9 pour CIC-IDS-2017.

Le processus de transformation comprend les étapes suivantes :

1. **Extraction et encodage des caractéristiques** : Suppression des colonnes liées aux étiquettes (`label`, `label_encoded`), ce qui donne une matrice de caractéristiques $X \in \mathbb{R}^{n \times m}$, où n est le nombre d'échantillons et m le nombre de caractéristiques. Pour NSL-KDD, les 41 caractéristiques initiales incluent des variables numériques (par exemple, `duration`, `src_bytes`) et catégoriques (par exemple, `protocol_type`, `service`, `flag`). Les colonnes catégoriques sont encodées en one-hot, augmentant le nombre de caractéristiques à $m = 122$. Par exemple :
   - `protocol_type` (3 valeurs : `tcp`, `udp`, `icmp`) génère 3 colonnes.
   - `service` (70 valeurs possibles) génère 70 colonnes.
   - `flag` (11 valeurs) génère 11 colonnes.

   Ainsi, les 41 caractéristiques initiales (38 numériques + 3 catégoriques) deviennent 122 après encodage one-hot (38 + 3 + 70 + 11). Pour CIC-IDS-2017, les 78 caractéristiques sont principalement numériques, avec un encodage minimal des catégoriques, maintenant $m = 78$.
2. **Normalisation** : Application de `StandardScaler` pour normaliser les caractéristiques à une moyenne nulle et une variance unitaire :

$$X_{\text{scaled}} = \frac{X - \mu}{\sigma} \tag{3.1}$$

   où μ et σ sont la moyenne et l'écart-type de chaque caractéristique.
3. **Calcul de la dimension de l'image** : Calcul de la longueur du côté d'une image carrée pour accueillir m caractéristiques :

$$\text{Side} = \lceil \sqrt{m} \rceil.$$

   Pour NSL-KDD ($m = 122$), $\sqrt{122} \approx 11,05$, donc côté $= 11$, créant une image de 11×11. Pour CIC-IDS-2017 ($m = 78$), $\sqrt{78} \approx 8,83$, donc côté $= 9$, créant une image de 9×9.

4. **Création des images** : Initialisation d'un tenseur d'images $X_{\text{img}} \in \mathbb{R}^{n \times \text{Side} \times \text{Side} \times 1}$. Pour chaque échantillon i, placement des m caractéristiques normalisées dans les m premières positions d'une image aplatie, avec un remplissage de zéros pour les positions restantes (par exemple, $121 - 122 = -1$ zéro pour NSL-KDD, $81 - 78 = 3$ zéros pour CIC-IDS-2017).

Par exemple, Considérons un jeu de données avec $m = 7$ caractéristiques par échantillon, par exemple, $[100, 500, 1, 0, 0, 1, 0]$ (représentant `duration`, `src_bytes`, `protocol_type_tcp`, etc.). Après normalisation, supposons que les valeurs deviennent $[0, 8, 1, 2, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0]$. La longueur du côté est :

$$\text{côté} = \lceil \sqrt{7} \rceil = \lceil 2,65 \rceil = 3.$$

L'image 3×3 est construite en plaçant les 7 caractéristiques dans les premières positions d'un tableau aplati, avec des zéros pour le remplissage :

$$\text{Aplati} : [0, 8, 1, 2, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0].$$

En remodelant en une matrice 3×3 :

$$\begin{bmatrix} 0,8 & 1,2 & 1,0 \\ 0,0 & 0,0 & 1,0 \\ 0,0 & 0,0 & 0,0 \end{bmatrix}.$$

Pour NSL-KDD ($m = 122$), l'image est de taille 11×11 (121 pixels), avec 122 caractéristiques normalisées (remplissage minimal). Pour CIC-IDS-2017 ($m = 78$), l'image est de taille 9×9 (81 pixels), avec 78 caractéristiques et 3 zéros de remplissage.

La transformation des données tabulaires en images permet d'utiliser les CNN pour capturer des motifs locaux et les Swin Transformers pour modéliser les dépendances globales, exploitant les hiérarchies spatiales malgré la nature non spatiale des données d'origine. Cette approche est essentielle pour appliquer le modèle hybride aux deux datasets.

3.4.2 Architecture du Modèle

Le modèle hybride CNN-Swin Transformer intègre les réseaux de neurones convolutifs (CNN) pour l'extraction de caractéristiques locales avec un bloc Swin Transformer pour la modélisation des dépendances globales. L'architecture traite les images en niveaux de gris dérivées des données de trafic réseau et est conçue pour gérer à la fois les tâches de classification binaire et multiclasse. Le tableau 3.1 résume les composants du modèle, y compris le nombre de couches, de filtres ou de neurones, et les paramètres clés.

Composant	Couches	Filtres	Détails
Couche d'Entrée	1	–	Forme d'entrée : (11,11,1) pour NSL-KDD ou (9,9,1) pour CIC-IDS-2017
CNN	2 convolutives	32, 64	Taille du filtre : 3x3, Activation : ReLU
CNN	1 pooling	–	Max pooling 2x2
Swin Transformer	1 bloc	–	Taille de la fenêtre : 2x2, 4 têtes d'attention, dimension de la tête : 64, inclut LayerNorm, MLP avec GELU, connexions résiduelles
Pooling Global Moyenne	1	–	Réduit les dimensions spatiales à un vecteur
Couches Denses	1 dense	128	Activation : ReLU, 30% dropout
Couche de Sortie	1	1 ou k	Sigmoid pour binaire (1 unité) ou Softmax pour multiclasse ($k = 5$ pour NSL-KDD, $k = 13$ pour CIC-IDS-2017)

TABLE 3.1 – Architecture du Modèle CNN-Swin Transformer

L'architecture est structurée comme suit :

- **Couche d'Entrée** : Le modèle accepte des images en niveaux de gris avec une forme (11,11,1) pour l'ensemble de données NSL-KDD ou (9,9,1) pour l'ensemble de données CIC-IDS-2017. Ces images sont générées à partir de caractéristiques de trafic réseau normalisées disposées dans une grille.
- **Couches Convolutives** :
  - La première couche convolutive (Conv2D) utilise 32 filtres avec un noyau 3x3 et une activation ReLU pour extraire des motifs locaux, tels que des corrélations entre des caractéristiques adjacentes (par exemple, octets source et destination).
  - Une couche de pooling maximum (MaxPooling2D) avec une taille de pool 2x2 réduit les dimensions spatiales, améliorant la robustesse aux petites variations de l'entrée.
  - La deuxième couche convolutive (Conv2D) applique 64 filtres avec un noyau 3x3 et une activation ReLU pour capturer des motifs plus complexes.

  La figure suivante illustre le flux de traitement du composant du réseau de neurones convolutifs (CNN), mettant en évidence la manière dont les caractéristiques locales sont extraites des images d'entrée.

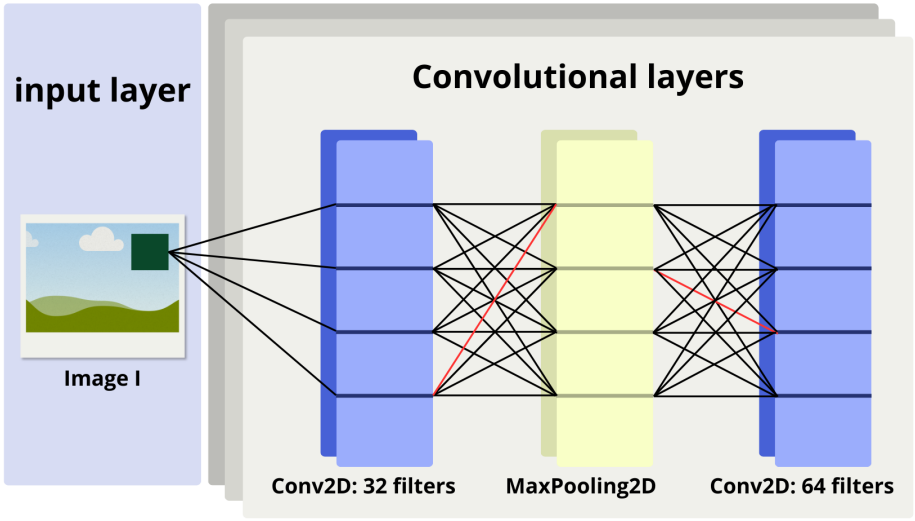


FIGURE 3.1 – Flux de traitement dans un réseau de neurones convolutifs (CNN)

- **Bloc Swin Transformer** : Un seul bloc Swin Transformer traite les cartes de caractéristiques pour modéliser les dépendances globales. Il comprend :
  - *Partitionnement des Fenêtres* : La carte de caractéristiques est divisée en fenêtres non chevauchantes de 2x2, chaque fenêtre étant traitée indépendamment.
  - *Normalisation des Couches* : Appliquée pour stabiliser l'entraînement en normalisant les activations.
  - *Attention Multi-Têtes* : Utilise 4 têtes d'attention, chacune avec une dimension de 64, pour capturer les relations au sein de chaque fenêtre.
  - *MLP* : Un perceptron multicouche à deux couches avec activation GELU pour les transformations non linéaires.
  - *Connexions Résiduelles* : L'entrée est ajoutée à la sortie des modules d'attention et MLP pour faciliter l'entraînement des réseaux profonds.
- **Pooling Global Moyenne** : Une couche de pooling global moyenne réduit les dimensions spatiales des cartes de caractéristiques à un seul vecteur en faisant la moyenne sur chaque canal, préparant les données pour les couches denses.
- **Couches Denses** :
  - Une couche dense avec 128 unités et activation ReLU combine les caractéristiques pour un raisonnement de haut niveau.
  - Une couche de dropout avec un taux de 30% est appliquée pour prévenir le surajustement.
- **Couche de Sortie** :
  - *Classification Multiclasse* : Une couche dense avec k unités ($k = 5$ pour NSL-KDD, $k = 13$ pour CIC-IDS-2017) et activation softmax sort les probabilités des classes.
  - *Classification Binaire* : Une couche dense avec 1 unité et activation sigmoid prédit la probabilité d'une attaque.

La figure suivante illustre le flux de traitement du bloc Swin Transformer combiné avec des couches denses, démontrant comment les dépendances globales sont modélisées et la classification est effectuée.

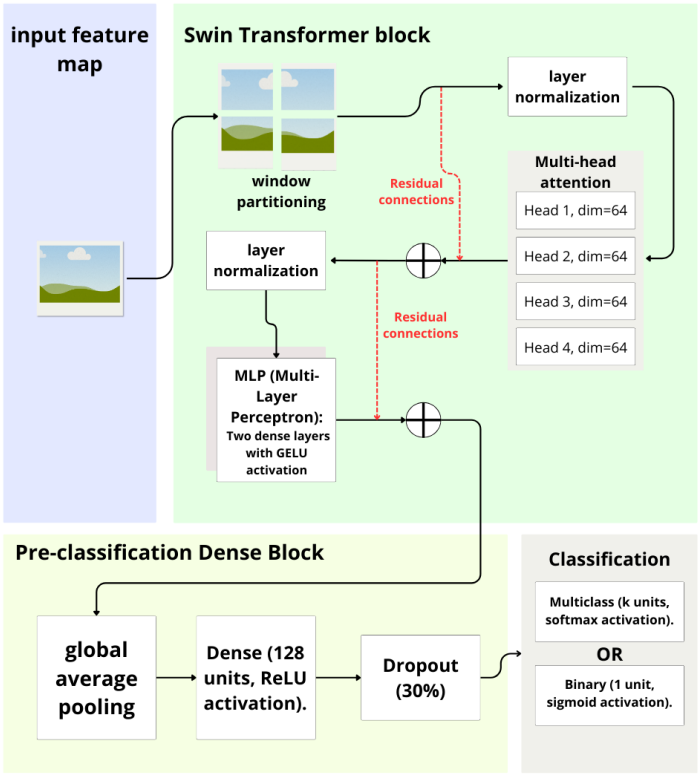


FIGURE 3.2 – Flux de traitement du modèle Swin Transformer avec des couches denses pour la classification

Cette architecture exploite la capacité du CNN à extraire des motifs locaux et celle du Swin Transformer à modéliser les relations globales, permettant une classification robuste et précise du trafic réseau.

3.4.3 Processus d'Entraînement

Le processus d'entraînement vise à optimiser le modèle hybride CNN-Swin Transformer pour minimiser l'erreur de classification sur les données d'intrusion réseau. Cela est réalisé par des mises à jour itératives des poids du modèle en utilisant la rétropropagation et l'optimisation basée sur le gradient. Le modèle est entraîné en utilisant des exemples étiquetés, et les données de validation sont utilisées pour surveiller les performances et prévenir le surajustement. Les mécanismes d'arrêt précoce aident à terminer l'entraînement une fois qu'aucune amélioration n'est observée sur plusieurs époques. Le processus d'apprentissage est guidé par la précision comme métrique de performance principale. Les détails complets de l'implémentation, y compris les outils logiciels et les paramètres des hyperparamètres, sont décrits dans le chapitre suivant.

3.4.4 Évaluation

Le modèle est évalué sur des ensembles de test, générant des prédictions comme suit :

- **Multiclasse** : Les probabilités sont converties en étiquettes de classe via argmax.
- **Binaire** : Les probabilités sont seuillées à 0,4 (NSL-KDD) ou 0,35 (CIC-IDS-2017).

3.4.5 Algorithme de l'Approche CNN-Swin Transformer

L'algorithme suivant décrit le processus de classification du trafic réseau en utilisant le modèle hybride CNN-Swin Transformer.

Algorithm 1 Algorithme de Classification du Trafic Réseau avec CNN-Swin Transformer

Entrée : Ensemble de données D (NSL-KDD ou CIC-IDS-2017)
Sortie : Métriques de performance pour la classification binaire et multiclasse
Préparation des Données
Charger D et extraire les caractéristiques F et les étiquettes L
if D est NSL-KDD **then**
    Encoder les caractéristiques catégorielles en utilisant l'encodage one-hot pour obtenir F (122 caractéristiques)
else
    F reste avec 78 caractéristiques (CIC-IDS-2017)
end if
Transformation en Images
Normaliser F avec StandardScaler pour obtenir F_{scaled}
Calculer la taille de l'image : side $= \lceil \sqrt{m} \rceil$ où m est le nombre de caractéristiques
if D est NSL-KDD **then**
    side $= 11$ (pour $m = 122$)
else
    side $= 9$ (pour $m = 78$)
end if
Créer des images I de taille side $\times$ side en plaçant F_{scaled} dans les premières positions et en remplissant le reste avec des zéros
Entraînement du Modèle
Initialiser le modèle hybride CNN-Swin Transformer M
Diviser les données en ensembles d'entraînement, de validation et de test avec un échantillonnage stratifié
Entraîner M sur I
Évaluation
Calculer les métriques : précision, précision, rappel, F1-score
Produire des rapports de classification par classe et la courbe ROC

3.4.6 Diagramme de l'Approche CNN-Swin Transformer

La figure suivante illustre le flux global de l'approche hybride CNN-Swin Transformer pour la classification du trafic réseau, détaillant le pipeline de la préparation des données à l'évaluation du modèle.

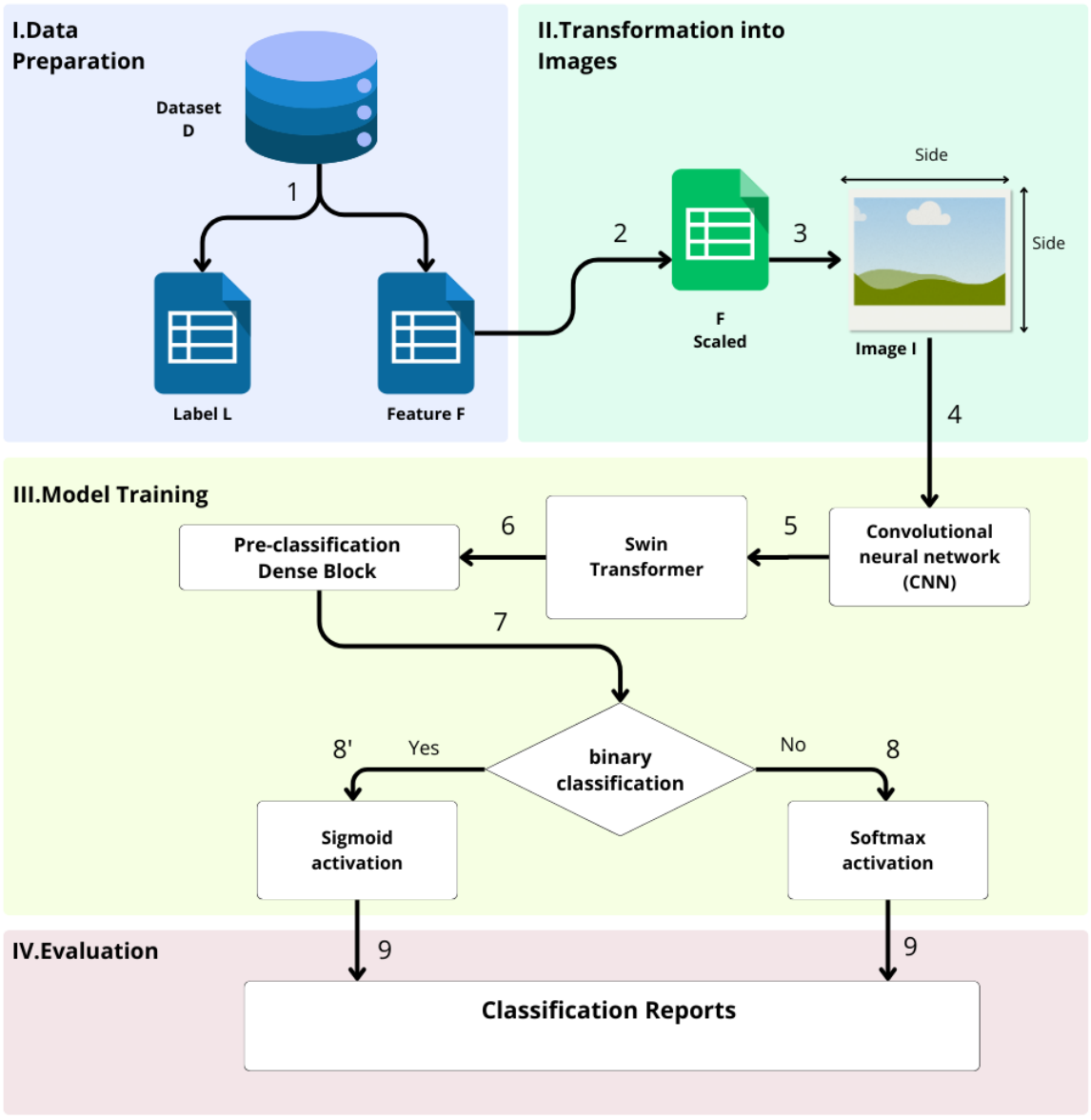


FIGURE 3.3 – Flux de l'approche hybride CNN-Swin Transformer pour la classification du trafic réseau.

Étapes de l'Approche CNN-Swin Transformer par Phase

Phase I. Préparation des Données

Etape 1 : Après avoir chargé l'ensemble de données D (ici NSL-KDD ou CIC-IDS-2017), extraire les caractéristiques F (Feature F) et les étiquettes L (Label L).

Etape 2 : Normaliser les caractéristiques F pour obtenir F_{scaled} (Scaled F), en utilisant la méthode *StandardScaler*.

Phase II. Transformation en Images

Etape 3 : Après avoir transformé en variables numériques les colonnes catégorielles si nécessaire, transformer F_{scaled} en images (Image I) de taille side $\times$ side (par exemple, 11x11 pour NSL-KDD, 9x9 pour CIC-IDS-2017), en plaçant les caractéristiques normalisées dans les premières positions et en remplissant avec des zéros si nécessaire.

Phase III. Entraînement du Modèle

Etape 4 : Traiter les images à l'aide d'un modèle hybride composé d'un réseau de neurones convolutifs (CNN) et d'un Swin Transformer, pour extraire et traiter les caractéristiques (CNN pour extraire les caractéristiques locales).

Etape 5 : Utiliser le Swin Transformer pour l'extraction des caractéristiques globales.

Etape 6 : Ajouter un bloc dense de pré-classification pour réduire les dimensions par pooling global, appliquer une transformation dense, puis préparer les données pour la classification.

Etape 7 : Appliquer la classification binaire (Oui) ou multiclasse (Non).

Etape 8 : Si classification binaire (Oui) : utiliser une activation sigmoïde (Activation Sigmoïde). Sinon, pour la classification multiclasse (Non) : utiliser une activation softmax (Activation Softmax).

Phase IV. Évaluation

Etape 9 : Générer des rapports de classification pour évaluer les performances du modèle.

3.5 Conclusion

Ce chapitre a permis de détailler la démarche suivie pour concevoir notre système de détection d'intrusions. En combinant les forces des Réseaux de Neurones Convolutifs (CNN) et du Swin Transformer, l'approche adoptée repose sur un enchaînement structuré de phases : de la préparation des données jusqu'à l'évaluation, en passant par leur transformation en images et l'entraînement du modèle. Cette organisation vise à exploiter au mieux les caractéristiques des données pour construire un modèle performant et adapté à la complexité des attaques à détecter.

Dans le chapitre suivant, nous procéderons à l'évaluation des performances du modèle proposé. Nous analyserons notamment les métriques de classification obtenues, comparerons les résultats avec ceux d'autres approches existantes, et discuterons de la robustesse et des limites de notre système.

Chapitre 4
Resultats et discussion

4.1 Introduction

Dans ce chapitre, nous présentons et discutons les résultats obtenus à partir de l'entraînement et de l'évaluation de notre modèle hybride CNN-Swin Transformer pour la détection d'intrusions réseau. Ce modèle a été conçu pour améliorer la précision et l'efficacité de la classification des attaques réseau en combinant les avantages des réseaux de neurones convolutifs (CNN) et des transformateurs, spécifiquement le Swin Transformer.

Nous commençons par décrire la configuration de l'entraînement, incluant les paramètres clés et les outils utilisés, tels que *Google Colab* avec un GPU T4, ainsi que les bibliothèques *TensorFlow* et et d'autres bibliothèques tel que : *pandas,numpy,scikit-learn,matplotlib,seaborn*. Ensuite, nous détaillons le processus de division des ensembles de données NSL-KDD et CIC IDS2017, en mettant l'accent sur les méthodes employées pour garantir une évaluation robuste et représentative des performances du modèle.

Les résultats de notre modèle sont présentés en deux sections principales : l'une pour l'ensemble de données NSL-KDD et l'autre pour CIC-IDS-2017. Pour chaque ensemble de données, nous évaluons les performances du modèle en termes d'exactitude, de précision, de rappel, de F1-score et de taux de faux positifs, tant pour la classification binaire que multiclasse. Nous incluons également des rapports de classification détaillés, des matrices de confusion et des courbes ROC pour fournir une vue d'ensemble complète des capacités du modèle.

Enfin, nous comparons les performances de notre modèle avec celles des méthodes existantes de l'état de l'art. Cette comparaison met en lumière les avantages et les limites de notre approche, tout en soulignant les contributions significatives de notre modèle hybride CNN-Swin Transformer à la détection d'intrusions réseau. Nous discutons également des points forts du modèle, de la synergie entre les CNN et le Swin Transformer, ainsi que des limites de l'approche proposée.

4.2 Configuration de l'Entraînement

Définition de Google Colab *Google Colab* est une plateforme *cloud* gratuite proposée par Google, permettant l'exécution de code Python directement dans un navigateur. Elle est largement utilisée pour l'entraînement de modèles en *machine learning* et *deep learning*, grâce à un accès sans frais à des ressources matérielles performantes, telles que des GPU (comme la NVIDIA T4) et des TPU (*Tensor Processing Units*) [60]. L'environnement s'exécute sur une machine virtuelle Linux (basée sur Ubuntu) hébergée par Google.

Ressources matérielles L'entraînement de notre modèle hybride a été effectué sur Google Colab avec un GPU NVIDIA T4 disposant de 16 Go de VRAM(Pour : *Video Random Access Memory*), environ 12,7 Go de RAM système, et 2 vCPUs(Pour : *Virtual Central Processing Unit*) (processeur Intel Xeon ou équivalent, fréquence ∼2,2 GHz). Le stockage temporaire est assuré par un disque SSD d'environ 70 Go.

Bibliothèques utilisées L'implémentation du modèle repose sur plusieurs bibliothèques Python :

- `pandas` et `numpy` pour la manipulation et le prétraitement des données.
- `scikit-learn` pour le prétraitement (normalisation avec `StandardScaler`, division des données avec `train_test_split`), le calcul des poids des classes (`compute_class_weight`),
  et l'évaluation des performances (`classification_report`, `confusion_matrix`, `roc_curve`, `auc`).
- `tensorflow` pour la construction, l'entraînement et l'évaluation du modèle de *deep learning* (via `tensorflow.keras`).
- `matplotlib` et `seaborn` pour la visualisation des résultats (courbes ROC, matrices de confusion).

Paramètres et performances Les principaux paramètres d'entraînement sont résumés dans le tableau 4.1. Pour le jeu de données CIC IDS2017 (119 323 échantillons), Le temps d'exécution a duré environ 562 secondes (∼9,4 minutes) pour un maximum de 100 époques, avec un arrêt précoce (patience de 10 époques) réduisant souvent à environ 60 époques. Pour le jeu de données NSL-KDD (125 965 échantillons), Le temps d'exécution a pris environ 289 secondes (∼4,8 minutes) pour un maximum de 30 époques, généralement 29 à 30 époques selon l'arrêt précoce. L'arrêt précoce optimise le temps d'entraînement en stoppant le processus lorsque l'exactitude n'évolue plus significativement sur 10 époques consécutives.

Paramètre	**Valeur**
Optimiseur	Adam
Taux d'Apprentissage	10^{-4}
Fonction de Perte	Entropie croisée binaire (binaire), Entropie croisée catégorielle éparse (multiclasse)
Taille du Lot	32 échantillons (NSL-KDD), 64 échantillons (CIC IDS2017)
Époques	Maximum 30 (NSL-KDD), Maximum 100 (CIC IDS2017)
Arrêt Précoce	Patience de 10 époques
Métrique Principale	Exactitude
Temps d'exécution	∼289 s (NSL-KDD), ∼562 s (CIC IDS2017)

TABLE 4.1 – Paramètres de Configuration de l'Entraînement

4.3 Division des Datasets

4.3.1 Processus de division

Le processus de division du jeu de données est effectué en deux étapes principales afin d'assurer une évaluation robuste des performances du modèle tout en préservant la distribution des classes grâce à une stratification. La division est réalisée à l'aide de la fonction `train_test_split` de la bibliothèque `scikit-learn`.

Dans un premier temps, le jeu de données est divisé en un ensemble d'entraînement complet et un ensemble de test, avec un ratio de 80 % pour l'entraînement complet et 20 % pour le test. Cette étape est formalisée comme suit :

```
X_train_full, X_test, y_train_full, y_test = train_test_split(
    X_img, y, test_size=0.2,
    stratify=y, random_state=42)
```

Dans un second temps, l'ensemble d'entraînement complet est divisé en un ensemble d'entraînement et un ensemble de validation, avec un ratio de 80 % pour l'entraînement et 20 % pour la validation. Cette étape est formalisée comme suit :

```
X_train, X_val, y_train, y_val = train_test_split(
    X_train_full, y_train_full, test_size=0.2,
    stratify=y_train_full, random_state=42)
```

L'utilisation du paramètre `stratify` dans les deux divisions garantit que la distribution des classes dans chaque sous-ensemble (entraînement, validation, test) reflète celle du jeu de données initial, ce qui est crucial pour les tâches de classification multi-classe avec des classes potentiellement déséquilibrées.

Les proportions finales des ensembles par rapport au jeu de données initial sont les suivantes :

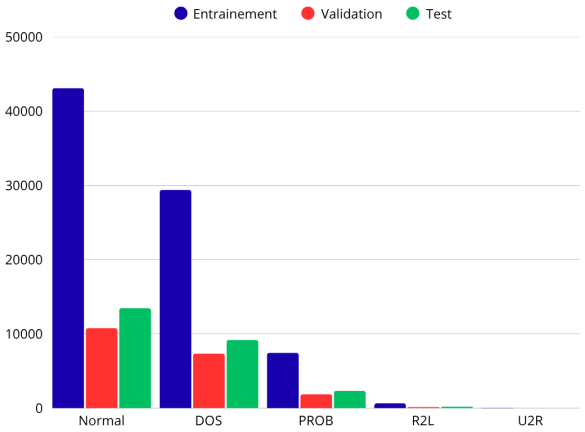
- **Ensemble d'entraînement** : 64 % du jeu de données initial.
- **Ensemble de validation** : 16 % du jeu de données initial.
- **Ensemble de test** : 20 % du jeu de données initial.

4.3.2 Nombre d'échantillons sur chaque Dataset

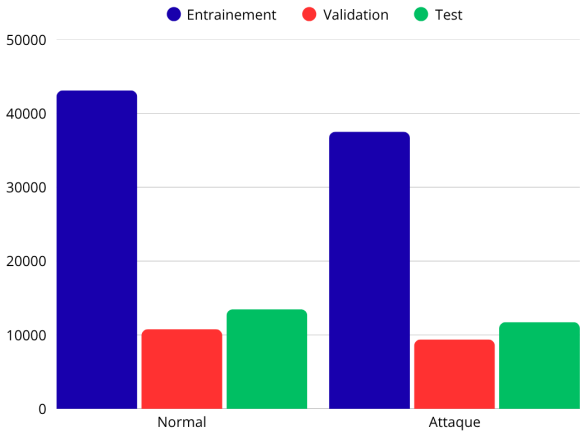
La répartition des échantillons dans les ensembles d'entraînement, de validation et de test est illustrée ci-dessous pour les datasets NSL-KDD et CIC IDS 2017.

Pour le dataset NSL-KDD, les graphiques suivants montrent le nombre d'échantillons par catégorie :

Pour le dataset CIC IDS 2017, les graphiques suivants présentent la distribution :

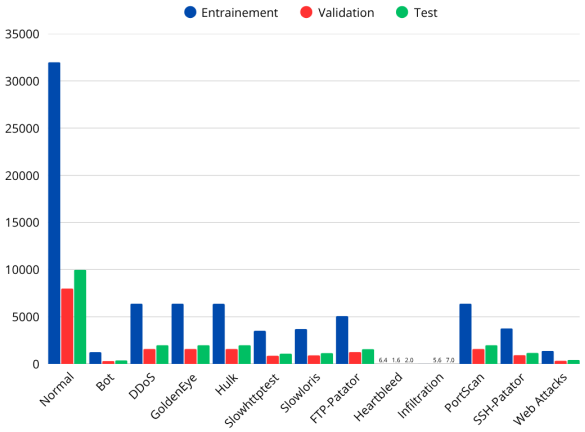


(a) Répartition des échantillons par catégorie détaillée pour NSL-KDD

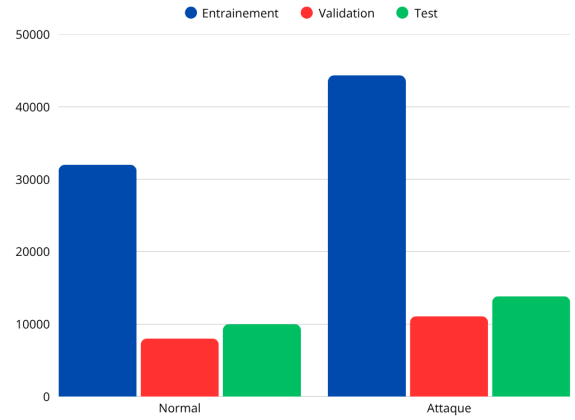


(b) Répartition des échantillons pour Normal et Attaque pour NSL-KDD

FIGURE 4.1 – Répartition des échantillons pour l'ensemble de données NSL-KDD



(a) Répartition des échantillons par catégorie détaillée pour CIC IDS 2017



(b) Répartition des échantillons pour Normal et Attaque pour CIC IDS 2017

FIGURE 4.2 – Répartition des échantillons pour l'ensemble de données CIC IDS 2017

4.4 Resultats sur : NSL-KDD

Nous évaluons d'abord les performances du modèle CNN-Swin Transformer sur l'ensemble de données NSL-KDD pour la classification binaire et multiclasse, comme le montre le tableau 4.2.

Métrique	**Classification Binaire**	**Classification Multiclasse**
Exactitude	99,69%	98,17%
Précision	99,69%	98,82%
Rappel	99,69%	98,17%
F1-Score	99,69%	98,40%
FPR	0,28%	-

TABLE 4.2 – Comparaison des Résultats de Classification pour NSL-KDD

Le tableau 4.2 montre que, pour la classification binaire, le modèle atteint une exactitude, une précision, un rappel et un F1-score de 99,69%, avec un taux de faux positifs (FPR) de 0,28%. Pour la classification multiclasse, il enregistre une exactitude de 98,17%, une précision de 98,82%, un rappel de 98,17% et un F1-score de 98,40%. Ces métriques démontrent des performances quasi parfaites dans les paramètres binaires et des résultats robustes dans les scénarios multiclasses, malgré les défis posés par les classes déséquilibrées.

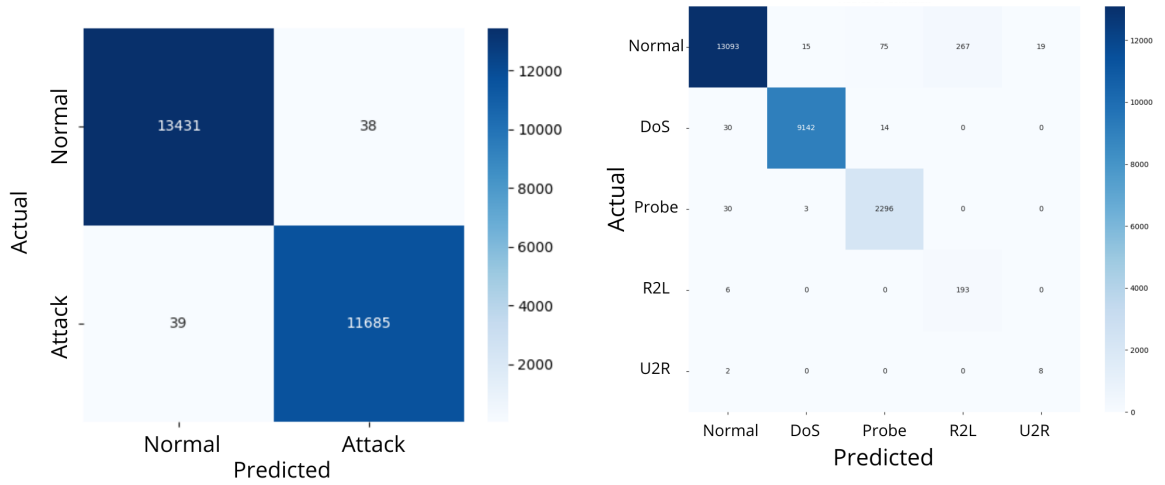
Nous présentons ensuite le rapport de classification détaillé pour la classification binaire sur l'ensemble de données NSL-KDD, comme détaillé dans le tableau 4.3.

Classe	Précision (%)	Rappel (%)	F1-Score (%)	Support
Normal	99,71	99,72	99,71	13469
Attaque	99,68	99,67	99,67	11724

TABLE 4.3 – Rapport de Classification pour NSL-KDD (Binaire)

Comme indiqué dans le tableau 4.3, le modèle identifie avec précision 99,71% des instances Normales et 99,67% des instances d'Attaque, démontrant une forte discrimination avec un minimum de mauvaises classifications.

Nous examinons maintenant les matrices de confusion pour la classification binaire et multiclasse sur l'ensemble de données NSL-KDD, comme illustré dans la figure 4.3.



(a) Matrice de Confusion pour la Classification Binaire (NSL-KDD)

(b) Matrice de Confusion pour la Classification Multiclasse (NSL-KDD)

FIGURE 4.3 – Matrices de Confusion pour l'Ensemble de Données NSL-KDD

La figure 4.3 confirme les performances robustes du modèle, avec un maximum de vrais positifs et de vrais négatifs et un minimum de faux positifs et de faux négatifs dans les classifications binaire et multiclasse, renforçant son efficacité dans la différenciation des classes.

Nous procédons maintenant, à l'évaluation du rapport de classification détaillé pour la classification multiclasse sur l'ensemble de données NSL-KDD, comme le montre le tableau 4.4.

Classe	Précision (%)	Rappel (%)	F1-Score (%)	Support
Normal	99,00	97,00	98,00	13469
DoS	100,00	97,00	99,00	9186
Probe	23,00	97,00	37,00	2329
R2L	42,00	97,00	59,00	199
U2R	30,00	80,00	44,00	10

TABLE 4.4 – Rapport de Classification pour NSL-KDD (Multiclasse)

Le tableau 4.4 révèle de bonnes performances pour les classes Normal (98% de F1-score) et DoS (99% de F1-score), mais des scores plus faibles pour les classes Probe (37% de F1-score), R2L (59% de F1-score) et U2R (44% de F1-score), reflétant les défis liés au support (Nombre d'echantillons pour le test) limité pour les types d'attaques rares (10 echantillons pour U2R et 199 pour R2L).

4.5 Resultats sur : CIC-IDS-2017

Nous analysons maintenant les performances du modèle CNN-Swin Transformer sur l'ensemble de données CIC-IDS-2017 pour la classification binaire et multiclasse, comme présenté dans le tableau 4.5.

Métrique	**Classification Binaire**	**Classification Multiclasse**
Exactitude	98,28%	96,87%
Précision	97,28%	97,30%
Rappel	99,82%	96,87%
F1-Score	98,54%	96,96%
FPR	0,18%	-

TABLE 4.5 – Comparaison des Résultats de Classification pour CIC-IDS-2017

Le tableau 4.5 indique que, pour la classification binaire, le modèle atteint une exactitude de 98,28%, une précision de 97,28%, un rappel de 99,82% et un F1-score de 98,54%, avec un FPR de 0,18%. Pour la classification multiclasse, il enregistre une exactitude de 96,87%, une précision de 97,30%, un rappel de 96,87% et un F1-score de 96,96%. Ces résultats mettent en évidence la capacité du modèle à détecter presque toutes les instances d'Attaque en classification binaire, avec une légère augmentation des faux positifs dans les paramètres multiclasses.

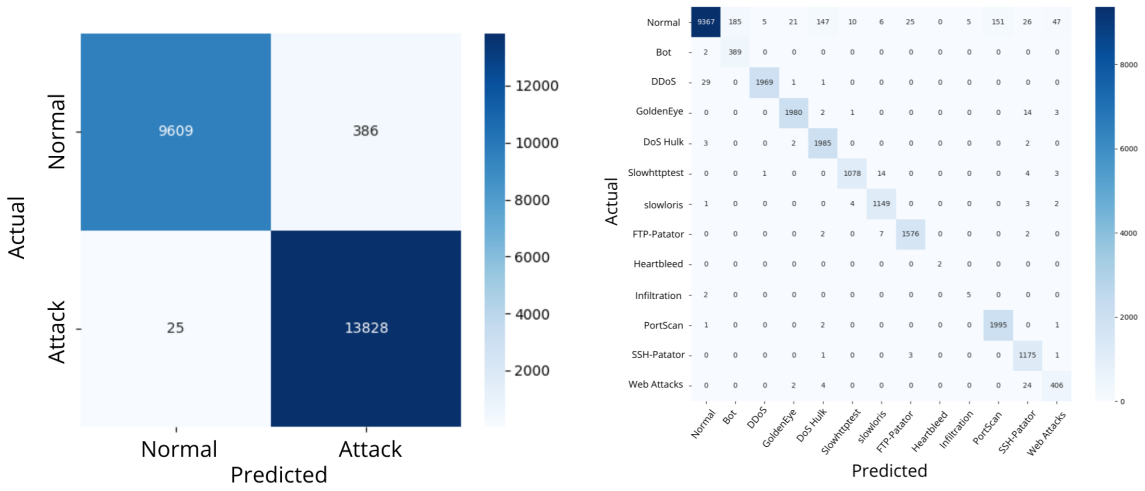
Nous examinons ensuite le rapport de classification détaillé pour la classification binaire sur l'ensemble de données CIC-IDS-2017, comme le montre le tableau 4.6.

Classe	Précision (%)	Rappel (%)	F1-Score (%)	Support
Normal	99,74	96,14	97,91	9995
Attaque	97,28	99,82	98,54	13853

TABLE 4.6 – Rapport de Classification pour CIC-IDS-2017 (Binaire)

Le tableau 4.6 montre un rappel élevé de 99,82% pour les instances d'Attaque, indiquant une excellente détection des activités malveillantes, bien que la précision de 97,28% suggère un petit nombre de faux positifs.

Nous explorons ensuite les matrices de confusion pour la classification binaire et multiclasse sur l'ensemble de données CIC-IDS-2017, comme illustré dans la figure 4.4.



(a) Matrice de Confusion pour la Classification Binaire (CIC IDS2017)

(b) Matrice de Confusion pour la Classification Multiclasse (CIC IDS2017)

FIGURE 4.4 – Matrices de Confusion pour l'Ensemble de Données CIC IDS2017

La figure 4.4 démontre la capacité du modèle à minimiser les faux positifs tout en maintenant des taux de détection élevés, en particulier en classification binaire. La matrice multiclasse montre de bonnes performances sur la plupart des classes, avec quelques défis pour les classes sous-représentées (ex. Hearthbleed et infiltration), notamment une confusion fréquente avec la classe dominante 'Normal' (ex. 185 instances de Bot predite comme Normal,151 instances de PortScan predite comme Normal...), et des erreurs inter-classes (ex. 14 instances de Slowhttptest prédites comme FTP-Patator)

Nous examinons en outre le rapport de classification détaillé pour la classification multiclasse sur l'ensemble de données CIC-IDS-2017, comme présenté dans le tableau 4.7.

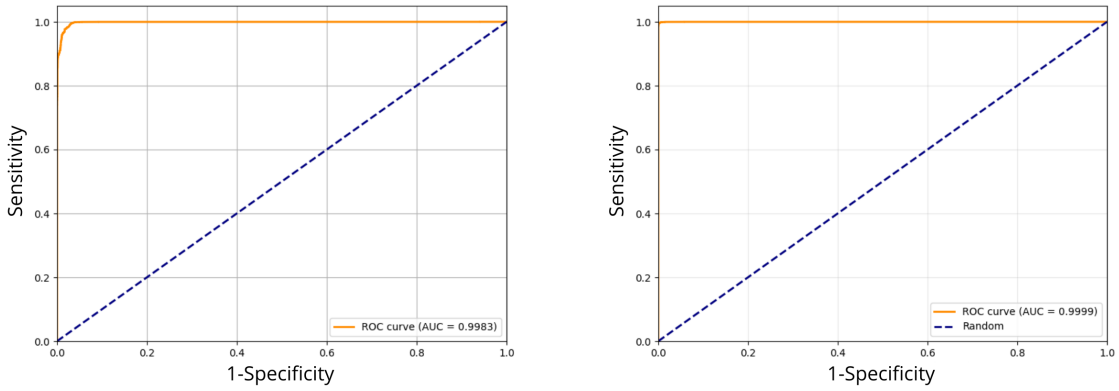
Classe	Précision (%)	Rappel (%)	F1-Score (%)	Support
Normal	100,00	94,00	97,00	9995
Bot	64,00	100,00	78,00	391
DDoS	100,00	100,00	100,00	2000
GoldenEye	99,00	99,00	99,00	2000
Hulk	93,00	100,00	96,00	1992
Slowhttptest	99,00	99,00	99,00	1100
slowloris	98,00	99,00	99,00	1159
FTP-Patator	99,00	99,00	99,00	1587
Heartbleed	100,00	100,00	100,00	2
Infiltration	50,00	86,00	63,00	7
PortScan	93,00	99,00	96,00	1999
SSH-Patator	95,00	99,00	97,00	1180
Web Attacks	89,00	91,00	90,00	436

TABLE 4.7 – Rapport de Classification pour CIC-IDS-2017 (Multiclasse)

Le tableau 4.7 met en évidence d'excellentes performances pour des classes comme DDoS et Heartbleed (100% de F1-score) et les variantes de DoS (99% de F1-score), mais des scores plus faibles pour Bot (78% de F1-score), Infiltration (63% de F1-score) et Web Attacks (90% de F1-score), indiquant des défis avec les motifs d'attaque sous-représentés ou complexes. Ces défis incluent une sous-représentation des données pour Infiltration (7 instances de test) et Heartbleed (2 instances de test), affectant la précision (ex. 50% pour Infiltration), une complexité des motifs pour Bot (78%) et Web Attacks (90%) due à des comportements variés, et un déséquilibre favorisant la classe Normal (9995 instances), entraînant des prédictions erronées pour les classes minoritaires.

4.6 Courbes ROC pour la classification binaire

Nous analysons ensuite les courbes ROC (En anglais : *Receiver Operating Characteristic*), une représentation graphique des performances d'un modèle de classification binaire pour tous les seuils de classification — où ROC signifie caractéristique de fonctionnement du récepteur (ou caractéristique de performance) [61] —, pour la classification binaire sur les ensembles de données NSL-KDD et CIC-IDS-2017, comme le montre la figure 4.5.



(a) Courbe ROC pour CIC IDS2017 (b) Courbe ROC pour NSL-KDD

FIGURE 4.5 – Courbes ROC pour la Classification Binaire

La figure 4.5 illustre le pouvoir discriminatoire fort du modèle, avec une AUC (*En anglais : Area Under the Curve*) de 0,999 pour NSL-KDD et 0,9983 pour CIC-IDS-2017, soutenant les valeurs de rappel élevé et de FPR (*En anglais : False Positive Rate*) faible rapportées précédemment. Ce pouvoir discriminatoire fort se manifeste par une valeur élevée de l'AUC, indiquant une excellente capacité à distinguer les classes Normale des Attaque sur tous les seuils, une séparation efficace des instances du trafic normal sur le trafic anormal(classe attaque) même dans des ensembles complexes comme CIC-IDS-2017, et une robustesse face aux déséquilibres potentiels des données, renforcée par un FPR faible(0.28% pour NSL-KDD et 0.18% sur CIC-IDS-2017).

4.7 Comparaison des Performances avec l'etat de l'art et discussions

4.7.1 Comparaison sur l'Ensemble de Données NSL-KDD

Nous comparons maintenant les performances de notre modèle CNN-Swin Transformer avec les méthodes existantes sur l'ensemble de données NSL-KDD, comme présenté dans le tableau 4.8.

Réf.	Méthode	Classi.	Acc.(%)	Préc.(%)	Rec.(%)	F1(%)
[38]	CNN-LSTM	Binaire	95	98	89	94
[45]	LSTM-TCN-LSTM	Binaire	N/A	N/A	97,68	98,00
[44]	CNN-BiLSTM	Binaire	95,96	97,21	99,12	94,60
[36]	IDS-RF	Binaire	98,67	98,68	98,67	98,67
[37]	ML Personnalisé	Binaire	97,5	99	96,7	95,7
Notre	**CNN-Swin Transformer**	**Binaire**	**99,69**	**99,69**	**99,69**	**99,69**
[46]	DS-kNN	Multiclasse	N/A	94,43	N/A	N/A
[47]	BERT-CNN-LSTM	Multiclasse	98,45	98	99	98,56
[43]	SMO-ANN	Multiclasse	99	N/A	N/A	N/A
[44]	CNN-BiLSTM	Multiclasse	98,42	N/A	N/A	N/A
[36]	IDS-RF	Multiclasse	98,54	97,58	96,29	96,92
[49]	Flow Transformer	Multiclasse	N/A	N/A	N/A	98,00
Notre	**CNN-Swin Transformer**	**Multiclasse**	98,17	**98,82**	98,17	98,40

TABLE 4.8 – Comparaison des Performances sur l'Ensemble de Données NSL-KDD

Le tableau 4.8 met en évidence les performances supérieures de notre modèle CNN-Swin Transformer en classification binaire, atteignant une précision, une précision, un rappel et un F1-score de 99,69%, surpassant des méthodes comme CNN-LSTM de Bamber et al. (95%) et IDS-RF d'Alshamy et Akcayol (98,67%). Notre approche évite la charge de calcul excessive liée à l'élimination récursive des caractéristiques utilisée par Bamber et al. [38] et utilise la pondération des classes pour gérer le déséquilibre entre les classes, sans avoir recours au suréchantillonnage, contrairement à certaines approches antérieures. Cela aboutit à un pipeline plus efficace et robuste pour les applications en temps réel. En classification multiclasse, notre modèle atteint une précision compétitive de 98,17 %, se rapprochant du modèle BERT-CNN-LSTM proposé par Ullah et al.[47] (98,45 %), et affichant de meilleures performances que des approches plus simples telles que IDS-RF (96,92 % de F1-score). Malgré ces bons résultats globaux, notre modèle présente encore certaines limitations concernant la détection des classes rares, notamment R2L (59 % de F1-score) et U2R (44 % de F1-score), en raison du faible nombre d'exemples représentatifs dans le jeu de données. Comme l'illustre le tableau 4.4, l'utilisation de la pondération des classes permet toutefois d'atténuer ce déséquilibre sans recourir à la génération de données synthétiques, contrairement à des méthodes comme DS-kNN de Taguelmimt et al [46], qui obtient de moins bons résultats sur les classes rares (94,43 % de précision sur KDD Cup 99).

4.7.2 Comparaison sur l'Ensemble de Données CIC-IDS2017

Enfin, nous comparons les performances de notre modèle avec d'autres méthodes sur l'ensemble de données CIC-IDS-2017, comme le montre le tableau 4.9.

Réf.	Méthode	Classi.	Acc.(%)	Préc.(%)	Rec.(%)	F1(%)
[45]	CNN-TCN-LSTM	Binaire	99,99	100,00	N/A	N/A
[47]	BERT-CNN-LSTM	Binaire	99,32	N/A	N/A	N/A
[43]	SMO-ANN	Binaire	97	N/A	N/A	N/A
Notre	**CNN-Swin Transformer**	**Binaire**	98,28	97,28	**99,82**	**98,54**
[50]	Vision Transformer	Multiclasse	96,40	N/A	N/A	N/A
[39]	DNN	Multiclasse	94,61	80,85	84,60	84,60
[39]	LSTM	Multiclasse	97,67	94,96	95,95	93,55
[39]	CNN	Multiclasse	98,61	97,05	95,00	95,00
[40]	DBN	Multiclasse	N/A	88,70	99,70	94,00
[41]	DNN	Multiclasse	97,62	65,29	88,58	67,16
[42]	1D-CNN	Multiclasse	98,96	98,70	99,20	98,94
Notre	**CNN-Swin Transformer**	**Multiclasse**	96,87	97,30	96,87	96,96

TABLE 4.9 – Comparaison des Performances sur l'Ensemble de Données CIC-IDS-2017

Le tableau 4.9 montre que notre modèle CNN-Swin Transformer atteint une précision de classification binaire de 98,28 %, légèrement inférieure à CNN-TCN-LSTM d'Amara et al. (99,99 %) [45]. Cependant, CNN-TCN-LSTM s'appuie sur le suréchantillonnage pour traiter le déséquilibre des classes, ce qui peut introduire des *artefacts* de données synthétiques (des motifs artificiels non représentatifs des données réelles) et augmenter la complexité *informatique*, limitant ainsi son *passage à l'échelle*. En revanche, notre modèle utilise la pondération des classes, offrant une approche plus efficace et généralisable sans recours à l'augmentation des données(via SMOTE ou autre technique de sur-echantillonage).

Pour la classification multiclasse, notre modèle atteint une précision de 96,87 %, ce qui est compétitif par rapport au 1D-CNN de Qazi et al. (98,96 %) [42]. Notamment, 1D-CNN est limité à 5 classes, tandis que notre modèle en gère 13, couvrant une plus large variété d'attaques, incluant les attaques par force brute, les botnets, les attaques DDoS, les infiltrations, entre autres. Cette capacité à traiter un plus grand nombre de classes montre une généralisation accrue face à la diversité des menaces, ce qui est essentiel dans un contexte de cybersécurité dynamique où les vecteurs d'attaque évoluent constamment.

De même, l'approche DBN de Belarbi et al. [40] prend en charge uniquement 6 classes et atteint un F1-score plus faible (94 %) par rapport à notre modèle (96,96 %) qui couvre un spectre plus large de classes. Comparé à d'autres méthodes comme le Vision Transformer de Ho et al. (96,40 %) [50] ou le DNN de Jose et al. (94,61 %) [39], notre modèle repose sur une architecture plus simple, combinant un CNN pour l'extraction de caractéristiques locales avec un Swin Transformer pour la modélisation des dépendances globales. Cette architecture hybride permet de conserver une complexité linéaire ($O(n)$), assurant une *efficacité accrue* au niveau de l'utilisation des ressources matérielles, tout en garantissant des performances robustes sur divers ensembles de données(NSL-KDD et CIC-IDS-2017).

4.8 Analyse du modèle hybride CNN-Swin Transformer

Le modèle hybride CNN-Swin Transformer, conçu pour la détection d'intrusions réseau, présente plusieurs points forts significatifs. Il atteint des performances remarquables, avec une exactitude de 99,69 % en classification binaire et 98,17 % en classification multiclasse sur l'ensemble NSL-KDD, ainsi que 98,28 % et 96,87 % respectivement sur CIC-IDS-2017. Ces résultats sont supérieurs aux résultats de plusieurs approches de l'état de l'art rapportées dans la littérature. L'approche évite le recours à des techniques de rééquilibrage des classes comme SMOTE, utilisant à la place une pondération adaptative des classes pour gérer les déséquilibres, ce qui réduit les biais artificiels. De plus, la transformation des données tabulaires en images en niveaux de gris permet d'exploiter efficacement les architectures de vision, rendant le modèle adaptable à des structures de données réseau complexes. Enfin, sa complexité linéaire ($O(n)$), assurée par l'utilisation du Swin Transformer, permet une réduction du temps de traitement et une meilleure utilisation des ressources que les modele transformers classique (ex.Vision Transformer), ce qui en fait une solution adaptée aux applications en temps réel.

La complémentarité entre CNN et Swin Transformer permet d'extraire simultanément des motifs locaux (via CNN) et des relations globales (grâce au mécanisme d'attention hiérarchique du Swin Transformer). Cette combinaison renforce la capacité de détection des attaques complexes tout en conservant une structure efficace sur le plan computationnel.

Malgré ses atouts, l'approche CNN-Swin Transformer présente certaines limites. Tout d'abord, elle montre des performances plus faibles pour les classes d'attaques rares, comme R2L (59 % F1-score) et U2R (44 % F1-score) sur NSL-KDD, ou Bot (78 %) et Infiltration (63 %) sur CIC-IDS-2017, en raison d'un fort déséquilibre des classes et d'un faible nombre d'exemples pour ces catégories. Ensuite, la transformation des données tabulaires en images peut parfois introduire du bruit ou entraîner une perte d'information, en particulier pour les attaques complexes aux motifs subtils. Par ailleurs, bien que le modèle présente une complexité linéaire avantageuse, le Swin Transformer reste relativement exigeant en ressources mémoire et en capacité de calcul lors de l'entraînement, ce qui peut poser problème dans des environnements fortement contraints, comme les dispositifs embarqués ou les déploiements sur périphériques IoT. Enfin, le modèle n'ayant pas été testé sur des scénarios dynamiques, tels que les attaques inconnues (*zero-day*) ou en apprentissage continu, sa capacité à généraliser dans ces contextes reste à évaluer.

4.9 Conclusion

Dans ce chapitre, nous avons présenté et analysé les résultats obtenus par notre modèle hybride CNN-Swin Transformer pour la détection d'intrusions réseau sur les ensembles de données NSL-KDD et CIC-IDS-2017. Les résultats montrent que notre modèle atteint des performances exceptionnelles, avec une exactitude de 99,69 % pour la classification binaire et 98,17 % pour la classification multiclasse sur NSL-

KDD. Sur CIC-IDS-2017, le modèle a obtenu une exactitude de 98,28 % pour la classification binaire et 96,87 % pour la classification multiclasse. Ces résultats sont supérieurs à plusieurs approches de l'état de l'art, démontrant ainsi la pertinence de notre architecture.

L'utilisation de l'environnement Google Colab, équipé d'un GPU NVIDIA T4 (16 Go de mémoire), a permis l'entraînement efficace de modèles complexes tout en maintenant des temps d'exécution raisonnables. Nous avons tiré parti des bibliothèques TensorFlow et d'autres bibliothèques pour la construction, l'entraînement et l'évaluation du modèle. La division stratifiée des ensembles de données a garanti une évaluation équilibrée, tandis que les rapports de classification, les matrices de confusion et les courbes ROC ont offert une vision complète des performances obtenues.

La comparaison avec les méthodes existantes met en lumière les apports de notre approche. La combinaison des CNN, pour l'extraction de caractéristiques locales, et du Swin Transformer, pour la modélisation des dépendances globales, renforce la capacité du modèle à identifier des schémas complexes liés aux intrusions.

Cependant, certaines limites subsistent. Les performances sont moins satisfaisantes pour les classes peu représentées, comme *R2L* ou *Infiltration*, en raison du déséquilibre prononcé dans les données d'entraînement. De plus, bien que le Swin Transformer soit plus léger que les Transformers classiques, ses besoins en mémoire GPU et en puissance de calcul peuvent poser des contraintes dans des environnements à ressources limitées. Une autre limite réside dans le fait que le modèle n'a pas encore été testé sur des flux de données dynamiques ou des attaques inconnues (*zero-day*), ce qui empêche, à ce stade, de conclure sur sa capacité à généraliser à des scénarios en temps réel.

En résumé, le modèle hybride CNN-Swin Transformer constitue une contribution prometteuse à la détection d'intrusions réseau. Grâce à ses bons résultats et à sa capacité d'adaptation à des structures complexes de données, il offre une base solide pour le développement de systèmes IDS intelligents. Néanmoins, des améliorations restent nécessaires, notamment pour renforcer la détection des attaques rares et explorer la performance du modèle dans des contextes évolutifs.

Conclusion Générale

Ce travail présente une nouvelle approche hybride CNN-Swin Transformer pour la détection d'intrusion réseau, conçue pour distinguer avec précision entre le trafic réseau malveillant et bénin. Les évaluations sur les ensembles de données NSL-KDD et CIC-IDS-2017 démontrent des performances exceptionnelles, avec des précisions de classification binaire de 99,69% et 98,28%, et des précisions multiclasses de 98,17% et 96,87%, respectivement. Ces résultats, soutenus par des métriques robustes telles que la précision, le rappel, le F1-score et les courbes ROC, soulignent l'efficacité du modèle dans la classification du trafic réseau tout en traitant le déséquilibre des classes par des fonctions de perte pondérées. En transformant les données tabulaires en images, la méthode proposée exploite les forces des CNN pour l'extraction de caractéristiques locales et des Swin Transformers pour la modélisation des dépendances globales, offrant ainsi des performances supérieures à plusieurs approches de pointe.

Cependant, l'approche présente des performances plus faibles pour certaines classes d'attaques rares, telles que R2L (59% de F1-score) et U2R (44% de F1-score) sur NSL-KDD, et Bot (78% de F1-score) et Infiltration (63% de F1-score) sur CIC-IDS-2017. Ces limitations proviennent de plusieurs facteurs. Premièrement, le déséquilibre sévère des classes dans ces ensembles de données, avec un nombre limité d'échantillons pour les types d'attaques rares, restreint la capacité du modèle à généraliser efficacement pour ces catégories. Deuxièmement, la transformation des données tabulaires en images, bien qu'innovante, peut introduire des artefacts ou une perte de caractéristiques subtiles cruciales pour détecter des attaques complexes comme l'Infiltration ou les Web Attacks, qui présentent des motifs variables ou nuancés. Enfin, malgré la complexité linéaire ($O(n)$) du Swin Transformer, ses exigences en ressources peuvent poser des défis dans des environnements à ressources limitées, limitant potentiellement l'optimisation pour les classes sous-représentées.

Malgré ces défis, l'efficacité de calcul et l'adaptabilité de l'approche la rendent bien adaptée au déploiement dans le monde réel. Les recherches futures pourraient explorer l'intégration de l'apprentissage continu et des techniques de détection d'anomalies, dans le but de réduire la variance du modèle — c'est-à-dire limiter sa sensibilité aux données d'entraînement — et d'améliorer sa généralisation à travers divers scénarios d'attaque.

Bibliographie

[1] Ammar Khraisat and Ali Alazab. A critical review of intrusion detection systems in the internet of things : techniques, deployment strategy, validation strategy, attacks, public datasets and challenges. *Cybersecurity*, 4(1) :1–27, 2021.

[2] Karen Scarfone and Peter Mell. Intrusion detection and prevention systems. In *Handbook of Information and Communication Security*. Springer, 2010.

[3] Neil Desai. Intrusion prevention systems : The next generation of intrusion detection. *Journal of Network and Computer Applications*, 32(1) :1–10, 2009.

[4] Poulmanogo Illy. Les systèmes de détection d'intrusion (ids). Technical report, École de Technologie Supérieure, April 2018. Rapport de recherche pour le cours IFT6271 - Sécurité informatique.

[5] Karen Scarfone and Peter Mell. Guide to intrusion detection and prevention systems (idps). Special Publication 800-94, National Institute of Standards and Technology (NIST), 2007.

[6] Carel Adelard and Osric Penrod. Implementation of network security system using firewall technology and intrusion detection system (ids). *Idea : Future Research*, 1(3) :113–121, 2023.

[7] Dafei Wu. Research on network security defense model based on combination strategy of firewall and ips. *Forest Chemicals Review*, pages 324–331, 2021.

[8] Wikipédia. Zone démilitarisée (informatique). `https://fr.wikipedia.org/wiki/Zone_d%C3%A9militaris%C3%A9e_(informatique)`, 2025. Consulté le 24 mai 2025.

[9] Olumide Babatope Longe, Babatunde Lawal, and Ayobami Ibitola. Strategic sensor placement for intrusion detection in network-based ids. *I.J. Intelligent Systems and Applications*, 2014(02) :61–68, 2014.

[10] Mehmet Ali Aydin, Abdul Halim Zaim, and K. Gokhan Ceylan. A hybrid intrusion detection system design for computer network security. *Computers Electrical Engineering*, 35(3) :517–526, 2009.

[11] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. Survey of intrusion detection systems : techniques, datasets and challenges. *Cybersecurity*, 2(1) :1–22, 2019.

[12] 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.

[13] Usman Ahmed, Mamoona Nazir, Awais Sarwar, Talha Ali, El-Hadi M Aggoune, Talha Shahzad, and Muhammad Aamir Khan. Signature-based intrusion detection using machine learning and deep learning approaches empowered with fuzzy clustering. *Scientific Reports*, 15(1) :1726, 2025.

[14] Stuart Russell and Peter Norvig. *Artificial Intelligence : A Modern Approach*. Pearson, 4th edition, 2021.

[15] Ethem Alpaydin. *Introduction to Machine Learning*. MIT Press, 4th edition, 2020.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[17] Jian Xu, Hao Chen, and Yang Zhou. Deep learning for network intrusion detection : Advances and challenges. *Computers & Security*, 126 :103–115, 2023.

[18] Rajesh Gupta, Priya Sharma, and Sanjay Kumar. Machine learning for intrusion detection : A comprehensive survey. *Journal of Network and Computer Applications*, 231 :103–124, 2024.

[19] Xin Li, Yu Zhang, and Lei Wang. Unsupervised learning for zero-day attack detection in cybersecurity. *IEEE Transactions on Information Forensics and Security*, 19 :456–467, 2024.

[20] Astera Software. Qu'est-ce que le prétraitement des données ? définition, concepts, importance, outils (2025), March 2025. Consulté le 19 avril 2025.

[21] Pure Storage. Qu'est-ce que le prétraitement des données pour l'apprentissage machine ?, April 2024. Consulté le 19 avril 2025.

[22] Jason Brownlee. How to split data into train and test sets with python. `https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/`, 2020. Consulted in 2025.

[23] Allen H. Renear, Simone Sacchi, and Karen M. Wickett. Definitions of dataset in the scientific and technical literature. *Proceedings of the American Society for Information Science and Technology*, 47(1) :1–4, 2010.

[24] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. NSL-KDD dataset for network-based intrusion detection systems. `https://www.unb.ca/cic/datasets/nsl.html`, 2009. Available at Canadian Institute for Cybersecurity, University of New Brunswick.

[25] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, pages 108–116. SciTePress, 2018.

[26] Salvatore J. Stolfo, Wenke Fan, Wenke Lee, Andreas Prodromidis, and Philip K. Chan. Kdd cup 1999 data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999. Dataset available at UCI KDD Archive.

[27] Nour Moustafa and Jill Slay. UNSW-NB15 : A comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). https://research.unsw.edu.au/projects/unsw-nb15-dataset, 2015. Available at Australian Centre for Cyber Security (ACCS), UNSW Canberra.

[28] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. CSE-CIC-IDS2018 dataset. https://www.unb.ca/cic/datasets/ids-2018.html, 2018. Available at Canadian Institute for Cybersecurity, University of New Brunswick.

[29] Patrik Goldschmidt and Daniela Chudá. Network intrusion datasets : A survey, limitations, and recommendations. *Computers & Security*, 2025. Preprint available on arXiv :2502.06688.

[30] Nicholas Pudjihartono, Tayaza Fadason, Andreas W. Kempa-Liehr, and Justin M. O'Sullivan. A review of feature selection methods for machine learning-based disease risk prediction. *Frontiers in Bioinformatics*, 2 :927312, June 2022. Open access article distributed under the terms of the Creative Commons Attribution License (CC BY).

[31] MonShotData. Métriques ml : Le guide ultime de l'évaluation des modèles, April 2025. Consulté le 01 juin 2025.

[32] Eugene H. Spafford and Diego Zamboni. Data collection mechanisms for intrusion detection systems. Technical Report 2000-08, CERIAS, Purdue University, June 2000.

[33] Data Science UA. Feature (data science). https://data-science-ua.com/wiki/data/feature/, 2023. Consulté le 05 juin 2025.

[34] Tcpdump Team. Tcpdump : A Packet Analyzer. http://www.tcpdump.org, 2023. Consulté le 6 juin 2025.

[35] National Institute of Standards and Technology. An introduction to information security. Technical Report SP 800-12 Rev. 1, NIST, 2001.

[36] Reem Alshamy and Muhammet Ali Akcayol. Intrusion detection model using machine learning algorithms on nsl-kdd dataset. *International Journal of Computer Networks & Communications (IJCNC)*, 16(6) :75–88, November 2024.

[37] Mohammed Zakariah, Salman A. AlQahtani, Abdulaziz M. Alawwad, and Abdullilah A. Alotaibi. Intrusion detection system with customized machine learning techniques for nsl-kdd dataset. *Computers, Materials & Continua*, 77(3) :4025–4053, 2023.

[38] Sukhvinder Singh Bamber, Aditya Vardhan Reddy Katkuri, Shubham Sharma, and Mohit Angurala. A hybrid cnn-lstm approach for intelligent cyber intrusion detection system. *Computers & Security*, 148 :104146, 2025. Available online 9 October 2024.

[39] Jinsi Jose and Deepa V. Jose. Deep learning algorithms for intrusion detection systems in internet of things using cic-ids 2017 dataset. *International Journal of Electrical and Computer Engineering (IJECE)*, 13(1) :1134–1141, February 2023. Open Access under CC BY-SA license.

[40] Othmane Belarbi, Aftab Khan, Pietro Carnelli, and Theodoros Spyridopoulos. An intrusion detection system based on deep belief networks. July 2023. Preprint submitted to arXiv.

[41] Gopichand Bandarupalli. Efficient deep neural network for intrusion detection using cic-ids-2017 dataset. *Research Square*, November 2024. Preprint.

[42] Emad Ul Haq Qazi, Abdulrazaq Almorjan, and Tanveer Zia. A one-dimensional convolutional neural network (1d-cnn) based deep learning system for network intrusion detection. *Applied Sciences*, 12(16), 2022.

[43] Deepshikha Kumari, Abhinav Sinha, Sandip Dutta, and Prashant Pranav. Optimizing neural networks using spider monkey optimization algorithm for intrusion detection system. *Scientific Reports*, 14(17196), July 2024.

[44] Rachid Ben Said, Zakaria Sabir, and Iman Askerzade. Cnn-bilstm : A hybrid deep learning approach for network intrusion detection system in software-defined networking with hybrid feature selection. *IEEE Access*, 11 :138732–138746, December 2023.

[45] Marwa Amara, Nadia Smairi, and Mohamed Jaballah. Stacked ensemble deep learning for robust intrusion detection in iot networks. In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025)*, volume 2, pages 1146–1153, Porto, Portugal, February 2025. SciTePress.

[46] Redha Taguelmimt and Rachid Beghdad. Ds-knn : An intrusion detection system based on a distance sum-based k-nearest neighbors. *International Journal of Information Security and Privacy*, 15(2) :131–144, April 2021.

[47] Farhan Ullah, Shamsher Ullah, Gautam Srivastava, and Jerry Chun-Wei Lin. Ids-int : Intrusion detection system using transformer-based transfer learning for imbalanced network traffic. *Digital Communications and Networks*, 10(1) :190–204, February 2024.

[48] Hai Zhou, Haojie Zou, Wei Li, Di Li, and Yinchun Kuang. Hivit-ids : An efficient network intrusion detection method based on vision transformer. *Sensors*, 25(6) :1752, March 2025.

[49] Liam Daly Manocchio, Siamak Layeghy, Wai Weng Lo, Gayan K. Kulatilleke, Mohanad Sarhan, and Marius Portmann. Flowtransformer : A transformer framework for flow-based network intrusion detection systems. *Expert Systems with Applications*, 241 :122564, May 2024.

[50] Chi Mai Kim Ho, Kin-Choong Yow, Zhongwen Zhu, and Sarang Aravamuthan. Network intrusion detection via flow-to-image conversion and vision transformer classification. *IEEE Access*, 10 :97780–97794, Aug 2022.

[51] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer : Hierarchical vision transformer using shifted windows. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, 2021. Open Access version provided by the Computer Vision Foundation.

[52] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words : Transformers for image recognition at scale. *International Conference on Learning Representations (ICLR)*, 2020.

[53] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998.

[54] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification : A comprehensive review. *Neural Computation*, 29(9) :2352–2449, 2017.

[55] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. pages 1–6, 2017.

[56] Aamir Javaid, Qazi Niyaz, Weiqing Sun, and Mansoor Alam. A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, pages 21–26, 2016.

[57] Tianlong Tang, Yi Deng, and Zhenhua Huang. An intrusion detection system based on convolutional neural network. In *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pages 1–4, 2019.

[58] Weida Wang, Qiang Sheng, Haibin Li, and Xiaoqing Yin. Hast-ids : Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. *IEEE Access*, 6 :1792–1806, 2017.

[59] Jinlong Wang, Yao Zhang, and Jianfeng Zhang. Canids : A can intrusion detection system based on swin transformer. *Electronics*, 12(5) :1190, 2023.

[60] Google Research. Faq de google colaboratory. https://research.google.com/colaboratory/faq.html?hl=fr, 2024. Consulté le 8 juin 2025.

[61] DataTab. Roc curve, 2025. Consulté le 11 juin 2025.

Annexe : Code source du modèle CNN–Swin Transformer

```python
# Classification multi-classe Pour le dataset CIC-IDS-2017
#Avec commentaires Pour la classification binaire et le dataset NSL-KDD
#===PHASE01:Préparation des Données========
#Importation des bibliotheques necessaires
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.utils.class_weight import compute_class_weight
import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow.keras import layers, models
from sklearn.metrics import roc_curve, auc
#AVANT LE CHARGEMENT DU DATASET dans ce code source, j'ai proceder a la:
# a. Suppression des doublons :
#    - Objectif : Éviter que le modèle apprenne plusieurs fois la même information, ce qui fausserait
#      les performances et pourrait conduire à un surapprentissage.
#    - Cette étape permet  d obtenir  un ensemble  d  échantillons plus varié et représentatif.

# b. Encodage des étiquettes (ajout de la colonne 'label_encoded') :
#    - Objectif : Transformer les classes textuelles ou catégorielles en valeurs numériques,
#      indispensables pour l'entraînement d'un modèle de deep learning.
#    - Exemple : 'BENIGN' devient 0, 'DoS Hulk' devient 1, etc.
#    - Cela permet  d utiliser  la classification multiclasse ou binaire avec les fonctions de perte adéquates.

# c. Réduction des classes majoritaires (uniquement sur CIC-IDS-2017) :
#    - Classe BENIGN limitée à 50 000 échantillons.
#    - Classes très fréquentes comme 'DoS Hulk', 'PortScan' réduites à 10 000.
#    - Objectif : Équilibrer manuellement le dataset pour éviter que le modèle ne favorise
#      les classes surreprésentées, surtout dans la classification binaire.
# === Spécifique au dataset NSL-KDD ===
# c. One-hot encoding des colonnes non numériques :
#    - Dans NSL-KDD, les colonnes 'protocol_type', 'service', 'flag' sont catégorielles.
#    - On utilise pd.get_dummies() pour les transformer en vecteurs binaires.

# 1. Chargement et nettoyage
```

```
df = pd.read_csv('/content/echantillons_cic_ids_2017_encoded.csv')
df = df.replace([np.inf, -np.inf], np.nan) #Supprime les valeurs Infinies et NaN
df = df.dropna(subset=['label_encoded'])
df = df.dropna()

# Fusion des classes 12, 13, 14 en une seule (classe 12 web attack)
df['label_encoded'] = df['label_encoded'].replace({13: 12, 14: 12})#Etape non disponible sur le dataset NSL-KDD

# 2. Préparation des données
X = df.drop(columns=['label', 'label_encoded']).values#Division du dataset en un ensemble d'echantillons sans etiquettes
y = df['label_encoded'].values #Ensemble d'etiquette

# Normalisation
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
#Application du one hot encoding sur NSL KDD
#non_numeric_cols = df.select_dtypes(include=['object']).columns.drop(['label'])
#df = pd.get_dummies(df, columns=non_numeric_cols)


#===PHASE02:Transformations des Données en Images========

# Transformation en images carrées
n_features = X_scaled.shape[1]
side = int(np.ceil(np.sqrt(n_features)))
X_img = np.zeros((X_scaled.shape[0], side, side, 1))
for i in range(X_scaled.shape[0]):
    X_img[i].flat[:n_features] = X_scaled[i]

##===PHASE03:Entrainement du modele========

# Split stratifié
X_train_full, X_test, y_train_full, y_test = train_test_split(X_img, y, test_size=0.2, stratify=y, random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_train_full, y_train_full, test_size=0.2, stratify=y_train_full, random_state=42)

# 3. Swin Transformer Block
def window_partition(x, window_size):
    B, H, W, C = tf.shape(x)[0], tf.shape(x)[1], tf.shape(x)[2], tf.shape(x)[3]
    pad_h = tf.math.maximum(0, window_size - H % window_size)
    pad_w = tf.math.maximum(0, window_size - W % window_size)
    x = tf.pad(x, [[0, 0], [0, pad_h], [0, pad_w], [0, 0]])
    H, W = H + pad_h, W + pad_w
    x = tf.reshape(x, (B, H // window_size, window_size, W // window_size, window_size, C))
    x = tf.transpose(x, perm=[0, 1, 3, 2, 4, 5])
    return tf.reshape(x, (-1, window_size, window_size, C))

def window_reverse(windows, window_size, H, W, C):
    pad_h = tf.math.maximum(0, window_size - H % window_size)
```

```
    pad_w = tf.math.maximum(0, window_size - W % window_size)
    H_padded, W_padded = H + pad_h, W + pad_w
    B = tf.shape(windows)[0] // (H_padded * W_padded // window_size //
        window_size)
    x = tf.reshape(windows, (B, H_padded // window_size, W_padded //
        window_size, window_size, window_size, C))
    x = tf.transpose(x, perm=[0, 1, 3, 2, 4, 5])
    x = tf.reshape(x, (B, H_padded, W_padded, C))
    return x[:, :H, :W, :]

class SwinTransformerBlock(tf.keras.layers.Layer):
    def __init__(self, dim, window_size, num_heads, **kwargs):
        super().__init__(**kwargs)
        self.dim = dim
        self.window_size = window_size
        self.num_heads = num_heads

    def build(self, input_shape):
        self.norm1 = layers.LayerNormalization()
        self.attn = layers.MultiHeadAttention(num_heads=self.num_heads,
            key_dim=self.dim)
        self.norm2 = layers.LayerNormalization()
        self.mlp = tf.keras.Sequential([
            layers.Dense(self.dim * 4, activation='gelu'),
            layers.Dense(self.dim)
        ])
        super().build(input_shape)

    def call(self, x):
        shortcut = x
        H, W = tf.shape(x)[1], tf.shape(x)[2]
        x = self.norm1(x)
        windows = window_partition(x, self.window_size)
        windows = tf.reshape(windows, (-1, self.window_size * self.
            window_size, self.dim))
        attn_windows = self.attn(windows, windows)
        attn_windows = tf.reshape(attn_windows, (-1, self.window_size,
            self.window_size, self.dim))
        x = window_reverse(attn_windows, self.window_size, H, W, self.dim
            )
        x = shortcut + x
        return x + self.mlp(self.norm2(x))

# 4. Modèle CNN + Swin Transformer
n_classes = len(np.unique(y))
inputs = layers.Input(shape=(side, side, 1))
x = layers.Conv2D(32, (3, 3), activation='relu', padding='same')(inputs)
x = layers.MaxPooling2D((2, 2))(x)
x = layers.Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = SwinTransformerBlock(dim=64, window_size=2, num_heads=4)(x)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(128, activation='relu')(x)
x = layers.Dropout(0.3)(x)
outputs = layers.Dense(n_classes, activation='softmax')(x)#Activation='
    Sigmoid' Pour la classification binaire

```

```
model = models.Model(inputs, outputs)
model.compile(optimizer=tf.keras.optimizers.Adam(1e-4),
              loss='sparse_categorical_crossentropy',#loss='Binary cross-entropy' dans la classification binaire
              metrics=['accuracy'])
model.summary()

# 5. Poids des classes
class_weights = compute_class_weight('balanced', classes=np.unique(y_train), y=y_train)
class_weight_dict = dict(enumerate(class_weights))

# Surpondération manuelle si souhaitée
if 12 in class_weight_dict:
    class_weight_dict[12] *= 3.0

# Poids par échantillon
sample_weights_train = np.array([class_weight_dict[label] for label in y_train])
sample_weights_val = np.array([class_weight_dict[label] for label in y_val])
#Seulment pour la classification binaire:Regalage du seuil
# Prédictions
#y_pred = model.predict(X_test)
#y_pred_classes = (y_pred > 0.4).astype(int).flatten() et (y_pred > 0.35) Pour NSLKDD

# 6. Entraînement
earlystop = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    sample_weight=sample_weights_train,
    epochs=100,#30 pour NSLKDD
    batch_size=64,#32 Pour NSLKDD
    callbacks=[earlystop],
    verbose=1
)
#===PHASE04:Evaluation et Productione des rapports de classification ========

# 7. Évaluation
print("\nÉvaluation finale :")
loss, acc = model.evaluate(X_test, y_test)
print(f"Accuracy test : {acc:.4f}")

# Prédictions et rapport
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
print("\nRapport de classification :")
print(classification_report(y_test, y_pred_classes))

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

```

```
accuracy = accuracy_score(y_test, y_pred_classes)
precision = precision_score(y_test, y_pred_classes, average='weighted')
recall = recall_score(y_test, y_pred_classes, average='weighted')
f1 = f1_score(y_test, y_pred_classes, average='weighted')

print("\n=== Métriques globales ===")
print(f"Exactitude (Accuracy)       : {accuracy * 100:.2f}%")
print(f"Précision pondérée          : {precision * 100:.2f}%")
print(f"Rappel pondéré              : {recall * 100:.2f}%")
print(f"F1-score pondéré            : {f1 * 100:.2f}%")

# 8. Matrice de confusion
conf_mat = confusion_matrix(y_test, y_pred_classes)
plt.figure(figsize=(12, 10))
sns.heatmap(conf_mat, annot=True, fmt='d', cmap='Blues')
plt.title("Matrice de Confusion")
plt.xlabel("Prédit")
plt.ylabel("Véritable")
plt.savefig("confusion_matrix_multiclass.png")
plt.show()
##Etapes seulment pour la classification binaire##
#FNR et courbe ROC Pour la classification binaire
# Taux  d attaques  détectées comme normal (Faux négatifs sur attaque)
fnr_attack = fn / (fn + tp) if (fn + tp) > 0 else 0
print(f"\nTaux d'attaques mal détectées (Attack prédit comme Normal - FNR
    ) : {fnr_attack * 100:.2f}%")
# Calcul de la courbe ROC
fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)

# Tracer la courbe ROC
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'Courbe ROC (AUC = {
    roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('1 - Spécificité')  # Faux positifs
plt.ylabel('Sensibilité')      # Vrais positifs
plt.title('Courbe ROC binaire')
plt.legend(loc='lower right')
plt.grid(True)
plt.savefig("roc_curve_binaire_cic-ids2017.png")
plt.show()
```

Listing 4.1 – Modèle hybride CNN–Swin Transformer appliqué au jeu de données KDDCUP99 sans suréchantillonnage

Abstract / Résumé

Abstract

We propose a novel intrusion detection system combining CNN and Swin Transformer. Tabular data (NSL-KDD, CIC-IDS-2017) are converted into grayscale images. CNN captures local patterns, Swin captures global dependencies. Our method outperforms others without rebalancing, reaching up to 99.69% (binary) and 98.17% (multiclass).

Keywords : Intrusion Detection System, CNN, Swin Transformer, NSL-KDD, CIC-IDS-2017

Résumé

Nous proposons un système de détection d'intrusion innovant combinant CNN et Swin Transformer. Les données NSL-KDD et CIC-IDS-2017 sont transformées en images. Le CNN détecte les motifs locaux, le Swin les dépendances globales. Sans rééquilibrage, notre modèle atteint jusqu'à 99,69% (binaire) et 98,17% (multiclasse).

Mots-clés : Système de Détection d'Intrusion, CNN, Swin Transformer, NSL-KDD, CIC-IDS-2017