République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ MOHAMED BOUDIAF DE M'SILA

Faculté de Mathématiques et Informatique

Département d'Informatique

Mémoire présenté en vue de l'obtention du diplôme de

Master Académique en Informatique

Spécialité : Système d'information et génie logiciel

Présenté par : Mihoubi Samir

Sallem Takieddine

Intitulé

Détection d'anomalies dans les réseaux informatiques à l'aide du Big Data Analytics et de l'intelligence artificielle

Soutenu devant le jury composé de :

Nom et prénom	Établissement	Qualité
Nom et prénom	Université de M'sila	Président
Nom et prénom	Université de M'sila	Examinateur
Dr. Ali Dabba	Université de M'sila	Superviseur
Mohamed Mihoubi	Université Sorbonne Paris Nord	Co-Supervisor

Année universitaire : 2025/2026

Dédicace

Je dédie ce travail à mes chers parents, en signe de reconnaissance pour leur amour, leur soutien permanent et les sacrifices qu'ils ont consentis tout au long de mon parcours. Leur confiance, leurs prières et leurs encouragements ont constitué une source essentielle de force, de patience et de persévérance.

Je dédie également ce modeste travail à ma famille, pour sa présence constante, sa compréhension et son soutien dans les moments difficiles comme dans les moments de réussite. Que chacun trouve ici l'expression de ma profonde gratitude.

À mes amis, pour leur accompagnement, leurs conseils et les moments partagés durant ces années d'études. Leur présence a rendu ce parcours plus humain et plus motivant.

Enfin, je dédie ce travail à mon binôme, Sallem Takieddine, pour son sérieux, son engagement et l'esprit de collaboration dont il a fait preuve tout au long de la réalisation de ce projet.

Mihoubi Samir

Je dédie ce travail à mes chers parents, pour leur amour inconditionnel, leur patience et les sacrifices qu'ils ont toujours acceptés afin de me permettre d'avancer dans mes études. Leur soutien moral, leurs conseils et leur confiance ont été pour moi une motivation précieuse à chaque étape de ce parcours.

Je le dédie aussi à ma famille, qui m'a toujours entouré d'encouragements et de bienveillance. Sa présence m'a aidé à surmonter les difficultés et à poursuivre mes objectifs avec détermination.

À mes amis et à toutes les personnes qui m'ont soutenu, directement ou indirectement, je témoigne toute ma reconnaissance pour leur aide, leur présence et leurs paroles encourageantes.

Enfin, je dédie ce travail à mon binôme, Mihoubi Samir, pour sa collaboration, sa persévérance et les efforts partagés durant la préparation de ce projet de fin d'études.

Sallem Takieddine

Remerciements

Nous remercions avant tout Allah, le Tout-Puissant, de nous avoir accordé la santé, la force, la patience et la volonté nécessaires pour mener ce travail à son terme.

Nous tenons à exprimer nos plus sincères remerciements à notre encadrant, Dr. Ali Dabba, pour la qualité de son encadrement, sa disponibilité, ses orientations scientifiques et ses remarques constructives. Son accompagnement rigoureux nous a permis de mieux structurer notre réflexion, de consolider notre démarche méthodologique et d'améliorer la qualité du travail présenté.

Nous adressons également nos vifs remerciements à notre co-encadrant, M. Mohamed Mihoubi, pour son soutien, ses conseils avisés et ses observations pertinentes. Son aide a contribué à orienter notre démarche expérimentale, à améliorer l'analyse des résultats et à enrichir la présentation de ce manuscrit.

Nous remercions respectueusement les membres du jury pour avoir accepté d'évaluer ce travail, pour le temps consacré à sa lecture et pour les remarques et suggestions qu'ils apporteront à son amélioration.

Nos remerciements s'adressent aussi à l'ensemble des enseignants du Département d'Informatique de l'Université Mohamed Boudiaf de M'sila, pour la formation, les connaissances et les valeurs scientifiques qu'ils nous ont transmises tout au long de notre cursus universitaire.

Enfin, nous exprimons notre reconnaissance à nos familles, à nos amis et à toutes les personnes qui nous ont soutenus, encouragés et accompagnés, directement ou indirectement, durant la réalisation de ce travail.

الملخص

يعالج هذا العمل إشكالية كشف الشذوذ والتسللات في حركة مرور الشبكات بالاعتماد على تقنيات معالجة البيانات الضخمة ونماذج التعلم العميق. يهدف العمل إلى بناء سلسلة تجريبية منظمة وقابلة لإعادة الإنتاج، تسمح بالتمييز بين التدفقات العادية والتدفقات الخبيثة ضمن مجموعة البيانات CICIDS2017، مع احترام شروط التقييم الصارم وتفادي تسرب المعلومات بين مراحل التدريب والاختبار.

تعتمد المنهجية المقترحة على تنظيف ملفات CSV، الاحتفاظ بالخصائص الرقمية الصالحة فقط، استبعاد أعمدة التصنيف والمصدر من مدخلات النماذج، إنشاء واصف ثنائي للتمييز بين الحركة السليمة والخبيثة، ثم تقسيم البيانات إلى مجموعات تدريب وتحقق واختبار. كما تم تطبيق توحيد المقاييس بالاعتماد على بيانات التدريب فقط، ومعايرة عتبات اتخاذ القرار باستخدام بيانات التحقق. وقد تم اعتماد إطارين للتقييم: إطار زمني يحتفظ بجزء DDoS Friday للاختبار النهائي، وإطار عام قائم على تقسيم طبقي يغطي عدة عائلات من الهجمات.

تمت مقارنة عائلتين من النماذج. يعتمد نموذج المرمز التلقائي المزيل للضجيج على تعلم بنية حركة المرور السليمة واستعمال خطأ إعادة البناء كمؤشر للشذوذ. أما النموذج الهجين CNN-MLP فيجمع بين فرع تلافيفي أحادي البعد لاستخراج الأنماط المحلية من متجه الخصائص، وفرع متعدد الطبقات لنمذجة العلاقات العامة بين مختلف الخصائص. أظهرت النتائج أن النموذج الهجين يقدم أداءً أكثر ملاءمة كتصنيف إشرافي عند توفر تسميات موثوقة، في حين يبقى المرمز التلقائي أداة مفيدة للمراقبة التكميلية وكشف السلوكيات غير المعتادة. تؤكد هذه النتائج أهمية جودة التحضير المسبق للبيانات، اختيار مقاييس تقييم مناسبة للفئات غير المتوازنة، وتحليل الأخطاء من خلال الإيجابيات والسلبيات الكاذبة.

الكلمات المفتاحية: كشف التسلل، شذوذ الشبكات، CICIDS2017، البيانات الضخمة، التعلم العميق، CNN-MLP، المرمز التلقائي.

Abstract

This work addresses the problem of network intrusion and anomaly detection by combining Big Data-oriented processing with deep learning models. The main objective is to design a structured and reproducible experimental pipeline capable of distinguishing benign and malicious network flows in the CICIDS2017 dataset, while ensuring rigorous evaluation conditions and preventing data leakage between training, validation, and testing stages.

The adopted methodology includes CSV file cleaning, the selection of valid numerical flow features, the exclusion of label-related and source-identifying attributes from model inputs, the construction of a binary descriptor for benign and malicious traffic, and a strict se-

paration of training, validation, and test subsets. Feature scaling is fitted only on the training data, and decision thresholds are calibrated on validation data. Two complementary evaluation settings are considered : a temporal setting in which the Friday DDoS subset is reserved for the final test, and a global stratified setting covering multiple attack families.

Two model families are evaluated. The Denoising Autoencoder learns the structure of benign traffic and uses reconstruction error as an anomaly score. The Hybrid CNN-MLP model combines a one-dimensional convolutional branch, designed to capture local patterns in the feature vector, with a multilayer perceptron branch, designed to model global relationships between features. The results show that the hybrid model is more suitable as a main supervised classifier when reliable labels are available, whereas the autoencoder remains useful as a complementary component for anomaly monitoring and the detection of unusual behaviors. The analysis also highlights the importance of data preprocessing quality, leakage prevention, threshold calibration, suitable metrics for imbalanced classes, and the interpretation of false positives and false negatives.

Keywords : intrusion detection, network anomaly detection, CICIDS2017, Big Data, deep learning, CNN-MLP, autoencoder.

Résumé

Ce travail traite la problématique de la détection d'intrusions et d'anomalies dans le trafic réseau en combinant une logique de traitement orientée Big Data avec des modèles d'apprentissage profond. L'objectif principal est de concevoir une chaîne expérimentale structurée et reproductible, capable de distinguer les flux réseau bénins des flux malveillants dans le jeu de données CICIDS2017, tout en garantissant des conditions d'évaluation rigoureuses et en évitant les fuites d'information entre les phases d'entraînement, de validation et de test.

La méthodologie adoptée comprend le nettoyage des fichiers CSV, la sélection des caractéristiques numériques valides des flux, l'exclusion des colonnes liées aux labels et à l'origine des données des variables d'entrée, la construction d'un descripteur binaire pour le trafic bénin et malveillant, ainsi qu'une séparation stricte des ensembles d'entraînement, de validation et de test. La normalisation des caractéristiques est ajustée uniquement sur les données d'entraînement, tandis que les seuils de décision sont calibrés sur les données de validation. Deux cadres d'évaluation complémentaires sont considérés : un cadre temporel dans lequel la partie Friday DDoS est réservée au test final, et un cadre global stratifié couvrant plusieurs

familles d'attaques.

Deux familles de modèles sont évaluées. Le Denoising Autoencoder apprend la structure du trafic bénin et utilise l'erreur de reconstruction comme score d'anomalie. Le modèle Hybrid CNN-MLP combine une branche convolutive unidimensionnelle, destinée à extraire des motifs locaux dans le vecteur de caractéristiques, avec une branche MLP, destinée à modéliser les relations globales entre les caractéristiques. Les résultats montrent que le modèle hybride est plus adapté comme classifieur supervisé principal lorsque des labels fiables sont disponibles, tandis que l'autoencodeur reste utile comme composant complémentaire pour la surveillance des anomalies et la détection de comportements inhabituels. L'analyse met également en évidence l'importance de la qualité du prétraitement, de la prévention des fuites de données, du calibrage des seuils, du choix de métriques adaptées aux classes déséquilibrées et de l'interprétation des faux positifs et des faux négatifs.

Mots-clés : détection d'intrusions, anomalies réseau, CICIDS2017, Big Data, apprentissage profond, CNN-MLP, autoencodeur.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

Abréviation	Signification
AE	Autoencoder
AI / IA	Artificial Intelligence / Intelligence Artificielle
API	Application Programming Interface
BCE	Binary Cross Entropy
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSV	Comma-Separated Values
CUDA	Compute Unified Device Architecture
cuDNN	CUDA Deep Neural Network Library
DDoS	Distributed Denial of Service
DL	Deep Learning
DNS	Domain Name System
F1	F1-score
GPU	Graphics Processing Unit
HDFS	Hadoop Distributed File System
HIDS	Host-based Intrusion Detection System
HTTP	HyperText Transfer Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
ML	Machine Learning
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
NIDS	Network Intrusion Detection System
PR-AUC	Precision-Recall Area Under Curve
ReLU	Rectified Linear Unit
RDD	Resilient Distributed Dataset
ROC-AUC	Receiver Operating Characteristic Area Under Curve
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol

Introduction générale

La sécurité des réseaux informatiques constitue aujourd'hui un enjeu majeur pour les organisations reposant sur des infrastructures numériques. Les services web, la messagerie électronique, les plateformes cloud, les systèmes distribués et les objets connectés fonctionnent grâce à des échanges constants entre équipements hétérogènes. Cette interconnexion soutient la productivité et la disponibilité des services, mais elle accroît aussi la surface d'exposition aux menaces telles que les attaques par déni de service, le balayage de ports, l'exploitation de vulnérabilités, la compromission d'équipements ou le vol d'identifiants [1, 2, 3].

Les dispositifs classiques de protection, comme les pare-feux ou les politiques de contrôle d'accès, filtrent certaines communications mais ne suffisent pas à détecter tous les comportements anormaux au sein du trafic autorisé. Les systèmes de détection d'intrusions (*Intrusion Detection Systems*, IDS) jouent alors un rôle complémentaire : ils surveillent les activités réseau ou système afin de repérer des indices d'attaque et d'assister l'administrateur dans la prise de décision [3, 4].

Les méthodes fondées sur des signatures ou des règles prédéfinies restent efficaces face aux attaques connues, mais leur performance diminue lorsque les attaques sont nouvelles, modifiées ou noyées dans un trafic légitime volumineux. Cette limite motive l'utilisation de techniques d'apprentissage automatique et profond capables d'extraire des régularités à partir des données et de modéliser des relations complexes entre les caractéristiques du trafic [5, 11, 13].

L'étude porte sur le développement, la mise en œuvre et l'évaluation de modèles d'apprentissage profond dédiés à la détection d'anomalies réseau. Elle s'appuie sur le jeu de données *CICIDS2017*, référence utilisée pour l'évaluation des IDS, comportant plusieurs jours de trafic normal et malveillant couvrant différentes familles d'attaques [19,20]. Ce choix permet d'étudier deux protocoles expérimentaux distincts : un protocole temporel strict et un protocole global plus représentatif de la diversité des scénarios.

Problématique

La problématique centrale consiste à déterminer comment exploiter efficacement les caractéristiques numériques extraites du trafic réseau pour différencier les flux normaux des flux malveillants, tout en maîtrisant le taux de faux positifs et en conservant une bonne capacité de généralisation. Cette question dépasse la recherche d'une simple précision élevée :

un IDS doit détecter les attaques réelles, limiter le bruit des alertes inutiles et rester fiable lorsque les données de test diffèrent des données d'apprentissage [8, 9, 10].

Objectifs

L'objectif principal est de définir une méthodologie expérimentale rigoureuse pour la détection d'anomalies réseau à partir du jeu de données *CICIDS2017*. Les objectifs spécifiques sont les suivants :

- **Analyser** la structure du trafic réseau et formaliser le passage des paquets bruts aux flux exploitables par les modèles d'apprentissage ;
- **Préparer** les données via la suppression des valeurs invalides, l'harmonisation des labels et la prévention des fuites d'information entre apprentissage, validation et test ;
- **Concevoir** un *Denoising Autoencoder* capable d'apprendre le comportement normal et de signaler les anomalies au moyen de l'erreur de reconstruction [6, 16] ;
- **Développer** un modèle *Hybrid CNN-MLP* combinant l'extraction locale de motifs par CNN 1D et l'apprentissage global des relations par un MLP ;
- **Comparer** ces modèles selon deux protocoles expérimentaux : une phase stricte avec séparation temporelle et une phase globale avec partitionnement stratifié 75 %/25 % ;
- **Évaluer** les performances à l'aide de métriques adaptées aux classes déséquilibrées, en particulier la précision, le rappel, le F1-score, la *balanced accuracy*, les faux positifs et les faux négatifs.

Méthodologie générale

La démarche adoptée s'articule en plusieurs étapes progressives. Après la présentation du contexte réseau et des enjeux de sécurité, les fondements de l'apprentissage profond sont introduits afin de justifier les modèles retenus. L'approche expérimentale repose ensuite sur deux phases complémentaires. La première utilise une séparation temporelle stricte : certains jours servent à l'apprentissage et *Friday DDoS* est réservé au test final. La seconde fusionne les données disponibles, applique le nettoyage, puis réalise un split stratifié 75 %/25 % couvrant plusieurs familles d'attaques.

La conservation de ces deux protocoles renforce la fiabilité de l'évaluation. Le protocole strict mesure la généralisation sur une journée exclue de l'apprentissage, tandis que le protocole global estime la performance moyenne sur l'ensemble des scénarios. Cette dualité évite de tirer des conclusions à partir d'une seule division des données.

Organisation

Le manuscrit se compose de quatre chapitres :

- **Le premier chapitre** établit le cadre conceptuel en présentant les réseaux informatiques, les principes de cybersécurité, les systèmes IDS/IPS, le Big Data et le jeu de données *CICIDS2017*.
- **Le deuxième chapitre** développe les bases théoriques de l'apprentissage profond appliqué à la détection d'intrusions, notamment le MLP, le CNN 1D, l'autoencodeur débruiteur, le modèle hybride et les métriques d'évaluation.
- **Le troisième chapitre** détaille l'environnement matériel et logiciel, les protocoles expérimentaux, le prétraitement et l'implémentation conceptuelle des modèles.
- **Le quatrième chapitre** présente les résultats, compare les approches, analyse les erreurs et discute les limites de l'évaluation.

Chapitre 1 : Contexte, sécurité réseau et acquisition des données

1.1 Introduction

Les réseaux informatiques constituent aujourd'hui la base essentielle des services numériques modernes. Les échanges de fichiers, la messagerie électronique, les applications web, les plateformes cloud, les objets connectés ainsi que les systèmes distribués reposent sur des communications constantes entre équipements hétérogènes. Cette interconnexion généralisée a significativement accéléré l'accès à l'information, amélioré la disponibilité des services et facilité la collaboration entre utilisateurs, tout en étendant notamment la surface d'exposition aux menaces informatiques [1, 2, 3].

Le trafic réseau contemporain se caractérise par un volume important, une continuité temporelle et une forte hétérogénéité. Un même réseau transporte simultanément des communications légitimes, telles que la consultation de services web, l'envoi de courriels ou les transferts de fichiers, et des activités malveillantes comprenant les balayages de ports, les attaques par déni de service ou les tentatives d'exploitation de vulnérabilités. La principale difficulté réside dans la capacité à différencier automatiquement et en temps réel un comportement normal d'un comportement suspect, au sein de flux dont la complexité et la volumétrie continuent d'augmenter.

La sécurité réseau ne se limite pas à la prévention des accès non autorisés. Elle implique également une surveillance continue du trafic, la collecte et le traitement de données, une analyse comportementale et une détection précoce des anomalies. Avant de recourir aux modèles d'apprentissage automatique ou profond, il est indispensable de comprendre la structuration des communications, la capture des paquets, leur agrégation en flux réseau, ainsi que la transformation de ces flux en données tabulaires exploitables par des algorithmes de classification.

Ce chapitre établit les bases conceptuelles et techniques de cette étude. Il présente successivement les fondamentaux des réseaux informatiques, incluant architectures, protocoles, paquets, flux et capture PCAP, ainsi que les enjeux liés à la sécurité réseau comme l'évolution des attaques, les limites des systèmes IDS/IPS traditionnels et le rôle des données massives. Il conclut par une introduction au jeu de données CICIDS2017, détaillant sa structure, ses scénarios, son étiquetage et les opérations de prétraitement indispensables à l'expérimentation.

1.2 Réseaux informatiques

Un réseau informatique se définit comme un ensemble d'équipements interconnectés permettant l'échange de données selon des protocoles communs [1, 2]. Ces équipements, qui incluent les ordinateurs, les serveurs, les routeurs, les commutateurs, les pare-feu, les smartphones et les objets connectés, assurent la communication et le partage de ressources telles que les fichiers, les applications, les bases de données et les services en ligne.

Les réseaux se distinguent par leur étendue géographique et leur usage. Un réseau local (*LAN*) relie des équipements dans un périmètre limité, tel qu'un bâtiment ou un campus. Un réseau étendu (*WAN*) connecte des sites éloignés géographiquement. Internet illustre un réseau mondial interconnectant des milliards de machines, soulignant le rôle incontournable des réseaux dans l'infrastructure numérique actuelle.

L'évolution des usages a engendré une croissance exponentielle du trafic, qui ne se limite plus aux échanges internes d'une organisation. Désormais, il englobe les services cloud, les flux multimédias, les sauvegardes distantes, les connexions mobiles et les interactions automatisées entre systèmes. Cette complexité rend la surveillance du trafic d'autant plus ardue qu'il est à la fois massif, rapide et varié.

1.2.1 Architecture et protocoles

L'architecture d'un réseau décrit l'organisation logique et physique de ses composants ainsi que les modalités de communication. Elle comprend généralement des postes clients, des serveurs, des commutateurs, des routeurs, des passerelles, des pare-feu et des systèmes de surveillance, chacun jouant un rôle précis dans l'acheminement et la protection des données.

La communication entre machines repose sur des protocoles, c'est-à-dire des ensembles de règles définissant l'envoi, la réception, l'interprétation et le contrôle des données. Elle s'organise selon des modèles en couches, principalement le modèle OSI et le modèle TCP/IP [1, 2], qui décomposent les fonctions en niveaux d'abstraction distincts.

Le modèle OSI identifie sept couches : physique, liaison de données, réseau, transport, session, présentation et application. Le modèle TCP/IP, plus usité, regroupe ces fonctions en quatre couches : accès réseau, Internet, transport et application. Dans l'analyse du trafic réseau, les couches réseau et transport sont cruciales, car elles véhiculent les adresses IP, les numéros de port et les identifiants de protocole.

Le protocole IP identifie les machines via leurs adresses et assure l'acheminement des paquets. Le protocole TCP garantit la fiabilité des échanges par un contrôle d'ordre, une retransmission et une détection de pertes. Le protocole UDP privilégie la rapidité au détriment de la fiabilité. Ces protocoles produisent des informations techniques exploitées ultérieure-

ment pour l'analyse du trafic et la détection d'intrusions.

1.2.2 Trafic réseau : paquets, flux et five-tuple

Toute information transmise sur un réseau, qu'il s'agisse d'un message, d'une requête web, d'un fichier ou d'un échange applicatif, est fragmentée en unités appelées paquets. Chaque paquet contient une portion des données échangées ainsi que les informations nécessaires à son acheminement entre hôtes.

Un paquet se compose essentiellement d'un en-tête (*header*) et d'une charge utile (*payload*). L'en-tête regroupe les métadonnées indispensables au transport, telles que l'adresse IP source et destination, les ports source et destination, le protocole, la taille du paquet et les indicateurs de contrôle. La charge utile contient le contenu applicatif transmis.

Dans la détection d'intrusions, l'analyse s'appuie surtout sur les métadonnées et les statistiques de communication, car la charge utile peut être chiffrée, volumineuse ou sensible sur le plan de la confidentialité [4, 5].

Un flux réseau (*flow*) regroupe un ensemble de paquets appartenant à une même communication logique. Dans les outils d'analyse, un flux est identifié par cinq attributs fondamentaux constituant le *five-tuple* : l'adresse IP source, l'adresse IP destination, le port source, le port destination et le protocole, formalisés comme suit :

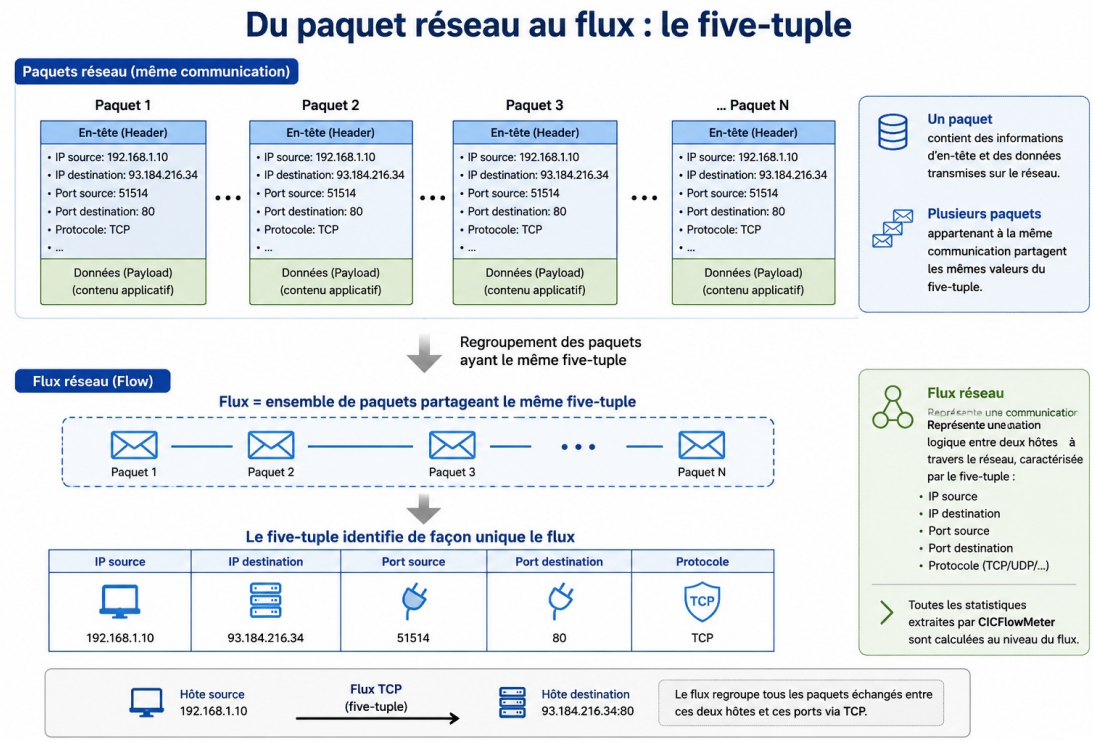

FIG. 1.1 : Passage des paquets réseau aux flux caractérisés par le five-tuple.

$$\text{Flow} = (IP_{\text{src}},\ IP_{\text{dst}},\ Port_{\text{src}},\ Port_{\text{dst}},\ \text{Protocole})\,. \tag{1.1}$$

La figure 1.1 illustre ce regroupement, où un flux rassemble plusieurs paquets partageant le même five-tuple. Cette représentation est essentielle puisque les modèles d'apprentissage traitent des caractéristiques statistiques extraites au niveau du flux, et non des paquets ni du contenu applicatif isolé.

Cette agrégation autorise l'étude du comportement global d'une communication : la durée, le nombre de paquets, le volume échangé, les tailles moyennes, le débit, la direction des échanges et les intervalles temporels entre paquets. Ces caractéristiques constituent les données d'entrée extraites par des outils comme CICFlowMeter et utilisées par les modèles d'apprentissage automatique et profond pour la détection d'intrusions.

1.2.3 Capture PCAP et extraction de flux avec CICFlowMeter

L'analyse du trafic débute par sa capture. Des outils tels que Wireshark ou d'autres *packet sniffers* interceptent les paquets circulant sur le réseau et les enregistrent dans des fichiers au format PCAP (*Packet CAPture*), fournissant une trace brute et exhaustive sur une période donnée.

Les modèles d'apprentissage automatique ne traitent pas directement les fichiers PCAP. Les paquets doivent être convertis en une représentation structurée. CICFlowMeter réalise cette étape en agrégeant les paquets en flux, puis en calculant pour chacun une série de caractéristiques statistiques.

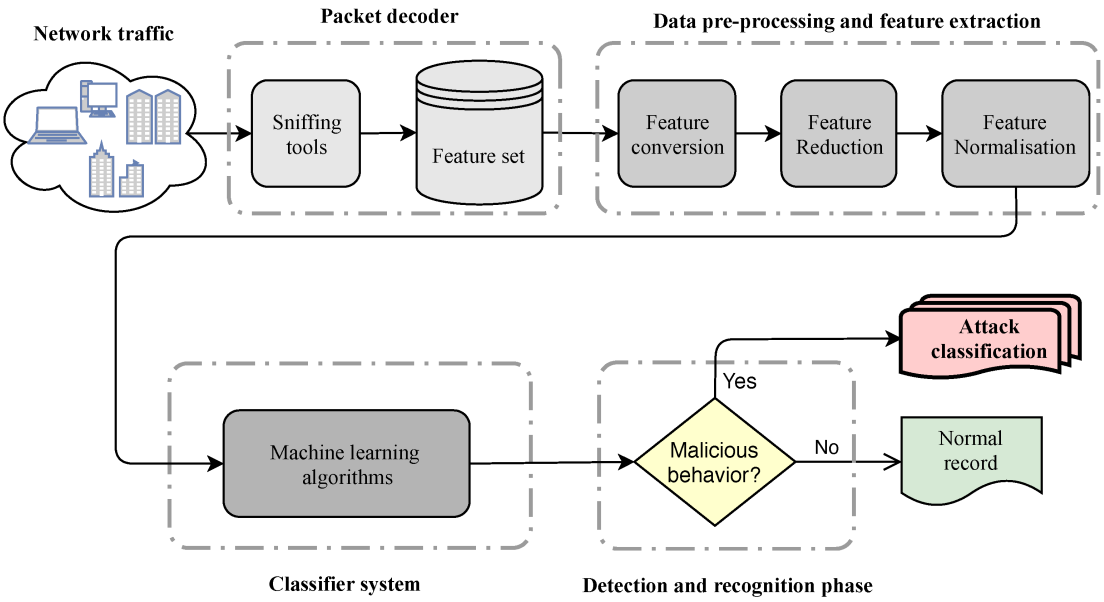

FIG. 1.2 : Passage des paquets bruts aux caractéristiques tabulaires exploitées par les modèles de détection d'intrusions.

Parmi ces caractéristiques figurent la durée du flux, le nombre de paquets envoyés et reçus, les tailles minimales, maximales et moyennes des paquets, le débit, les temps inter-arrivée, les indicateurs TCP et le volume de données transmises. Le résultat est un fichier CSV où chaque ligne correspond à un flux et chaque colonne à une caractéristique [19, 20].

Tab. 1.1 : Exemples de familles de caractéristiques extraites par CICFlowMeter.

Famille	Exemples	Intérêt pour un IDS
Durée et débit	`Flow Duration`, `Flow Bytes/s`, `Flow Packets/s`	Détecter les communications anormalement rapides, longues ou volumineuses.
Paquets avant/arrière	`Total Fwd Packets`, `Total Backward Packets`	Mesurer l'asymétrie entre requêtes et réponses.
Tailles de paquets	`Packet Length Mean`, `Max Packet Length`, `Fwd Packet Length Mean`	Repérer des profils volumétriques typiques de certaines attaques.
Intervalles temporels	`Flow IAT Mean`, `Fwd IAT Std`, `Bwd IAT Max`	Identifier des rythmes de communication réguliers, brusques ou automatisés.
Indicateurs TCP	`SYN Flag Count`, `ACK Flag Count`, `RST Flag Count`	Décrire les comportements de connexion, de scan ou d'interruption.

Ce tableau relie directement les colonnes numériques aux comportements réseau observables. Ainsi, une attaque DDoS peut se manifester par un débit élevé et de nombreux paquets, alors qu'un balayage de ports peut se traduire par des connexions courtes, répétitives et fortement marquées par certains indicateurs TCP.

La figure 1.2 schématise cette transformation, qui convertit le trafic brut en données tabulaires directement exploitables par les algorithmes d'apprentissage. Chaque flux devient une observation numérique, dont les caractéristiques servent à différencier le trafic normal du trafic malveillant.

1.3 Sécurité des réseaux

La sécurité des réseaux vise à garantir la protection des ressources, des communications et des données contre les accès non autorisés, les perturbations, les altérations illégitimes et les attaques [3]. Elle constitue un pilier fondamental de la cybersécurité, dans un contexte où la quasi-totalité des systèmes d'information repose sur des infrastructures interconnectées.

Un incident de sécurité réseau peut provoquer une interruption de service, une fuite de données confidentielles, une altération des informations, une saturation des ressources, une compromission d'équipements ou une propagation latérale d'attaques. La sécurité ré-

seau s'appuie donc sur des mécanismes complémentaires de prévention, de détection et de réaction.

1.3.1 Enjeux, surface d'attaque et modèle CIA

Les enjeux majeurs de la sécurité de l'information sont classiquement modélisés par le triptyque CIA :

- **Confidentialité** : vise à empêcher tout accès non autorisé aux données et aux communications.
- **Intégrité** : garantit que les données ne sont pas altérées illégalement, assurant leur exactitude et leur fiabilité.
- **Disponibilité** (*Availability*) : assure l'accès aux services pour les utilisateurs légitimes au moment opportun.

Dans un réseau moderne, ces trois dimensions peuvent être compromises de diverses manières :

- Une attaque passive par écoute (*sniffing*) compromet la **confidentialité**.
- La modification de paquets ou l'exploitation de vulnérabilités affectent l'**intégrité**.
- Les attaques par déni de service compromettent la **disponibilité**.

La notion de surface d'attaque désigne l'ensemble des points d'entrée exploitables par un attaquant. Plus un système expose de services, d'interfaces, d'utilisateurs ou d'équipements, plus sa surface d'attaque s'étend. L'émergence du cloud, de la mobilité et des objets connectés a considérablement augmenté cette surface, rendant indispensable une surveillance continue du trafic [4].

1.3.2 Taxonomie des attaques réseau

Les attaques réseau se catégorisent selon leur objectif, leur vecteur d'attaque ou leur impact sur la cible [3]. Les jeux de données de cybersécurité, comme CICIDS2017, regroupent typiquement plusieurs familles représentatives de la menace actuelle :

- Les **attaques par déni de service** (DoS) visent à rendre un service indisponible par saturation des ressources ou exploitation de failles.
- Les **attaques DDoS** (*Distributed DoS*) mobilisent des machines compromises réparties géographiquement, compliquant ainsi leur neutralisation.

- Les **PortScan** consistent à sonder les ports ouverts pour identifier les services actifs et préparer une attaque.
- Les attaques par **force brute** tentent de deviner des identifiants ou des mots de passe par énumération exhaustive.
- Les **attaques applicatives**, comme l'injection SQL ou le *Cross-Site Scripting* (XSS), ciblent les applications web exposées.
- Les **botnets** désignent des réseaux de machines compromises contrôlées à distance.

Chaque catégorie laisse des traces spécifiques dans le trafic :

- Une attaque DDoS génère un volume anormalement élevé de paquets.
- Le PortScan produit de nombreuses connexions éphémères vers divers ports.
- La force brute multiplie les tentatives d'authentification.

Ces signatures comportementales justifient l'usage de caractéristiques statistiques pour automatiser la reconnaissance des activités suspectes [6, 11].

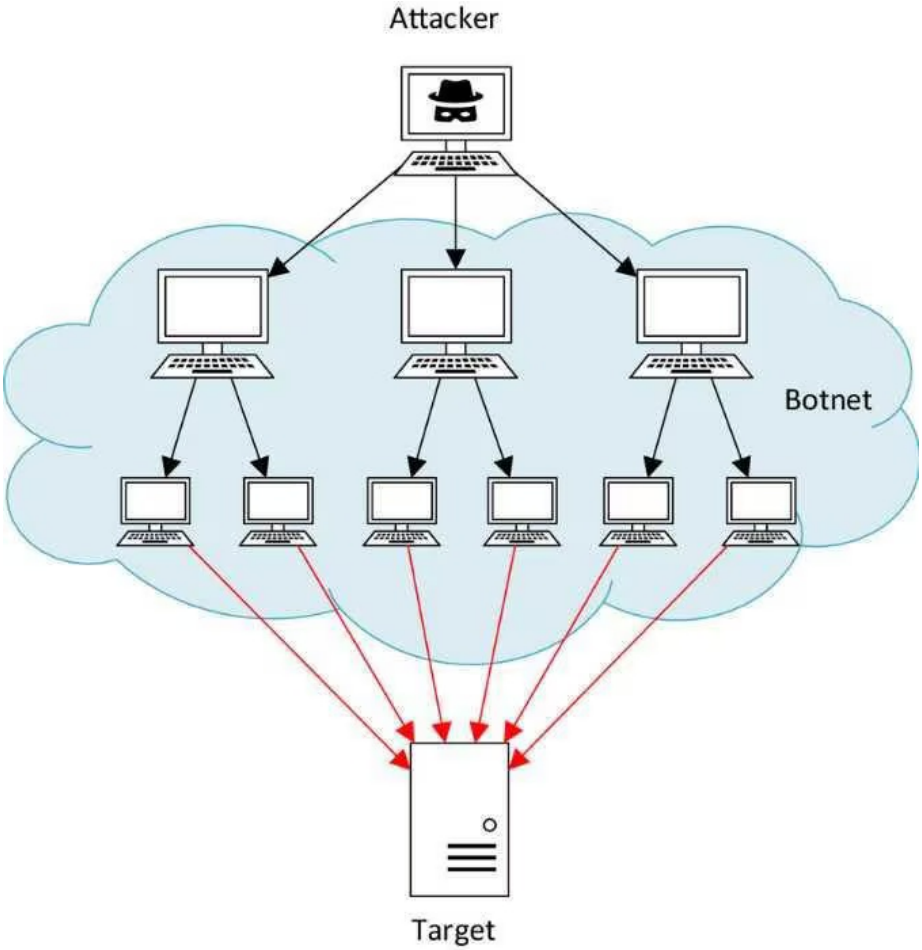

FIG. 1.3 : Exemple schématique d'une attaque DDoS : plusieurs machines compromises inondent simultanément une cible pour perturber sa disponibilité.

1.3.3 Systèmes IDS/IPS : approches et limites

Un système de détection d'intrusions (IDS) a pour fonction d'observer les activités réseau ou système afin d'identifier des comportements suspects et de déclencher des alertes. Un système de prévention d'intrusions (IPS) étend cette capacité en bloquant automatiquement les actions dangereuses. Ces dispositifs sont des éléments clés des architectures de sécurité défensive.

Deux grandes catégories sont classiques :

- Le **NIDS** (*Network IDS*) qui analyse le trafic réseau.
- Le **HIDS** (*Host IDS*) qui surveille les activités internes d'une machine, telles que les journaux système, les processus actifs, les appels système et les fichiers sensibles.

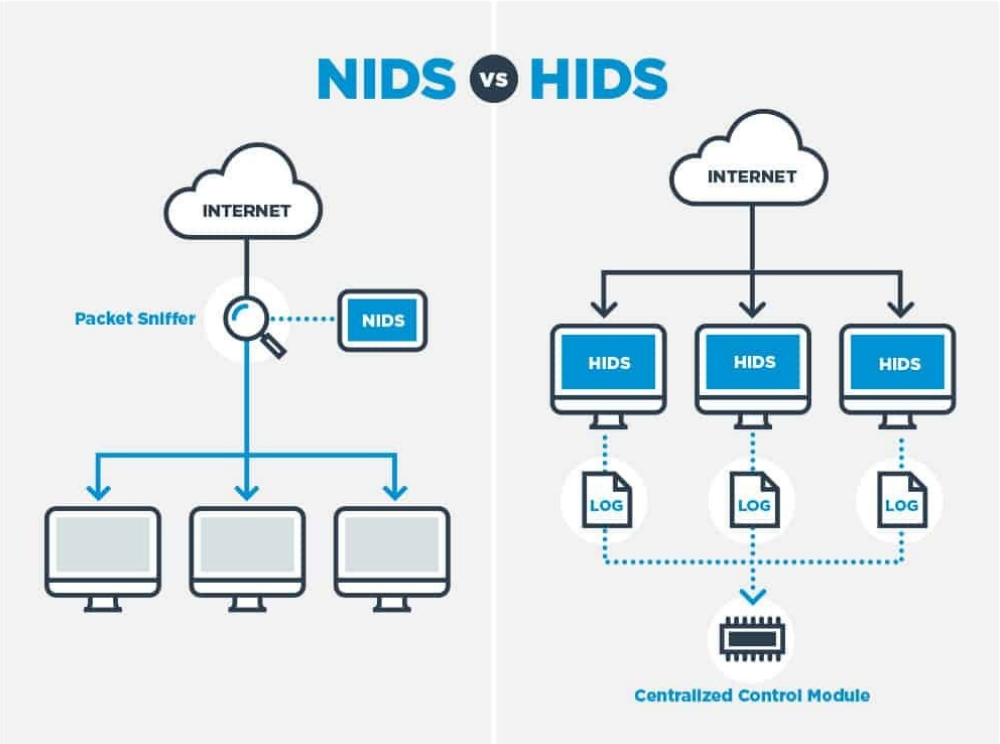

Fig. 1.4 : Distinction entre NIDS et HIDS au sein d'une architecture de surveillance de sécurité réseau.

Les systèmes IDS/IPS traditionnels reposent essentiellement sur des **signatures**, c'est-à-dire des motifs caractéristiques d'attaques déjà identifiées. Lorsqu'un trafic correspond à l'une de ces signatures, une alerte est déclenchée. Cette méthode s'avère efficace contre les menaces connues, mais montre ses limites face aux attaques émergentes, aux variantes inédites ou à celles spécifiquement conçues pour contourner les règles existantes. Les principales contraintes sont les suivantes :

- Les **faux positifs** : un trafic légitime est erronément classé comme malveillant, engendrant une surcharge d'alertes susceptibles d'épuiser les analystes et de masquer les menaces réelles.
- Les **faux négatifs** : des attaques non détectées, ce qui constitue un risque majeur en permettant à un attaquant d'opérer sans être repéré.
- Le **trafic chiffré** : empêche l'inspection directe de la charge utile, rendant nécessaire l'exploitation des métadonnées et des statistiques de flux, renforçant ainsi l'intérêt des méthodes basées sur l'apprentissage automatique.

1.4 Big Data et cybersécurité

Le trafic réseau actuel génère en continu des volumes de données considérables. Chaque paquet, connexion, événement ou journal système constitue une trace exploitable. Dans les infrastructures à grande échelle, ces flux dépassent largement la capacité des méthodes d'analyse classiques, confrontant la cybersécurité à la problématique des données massives, communément désignée sous le terme de *Big Data* [23, 24, 25].

Le Big Data ne se définit pas uniquement par la quantité, mais par la nécessité d'utiliser des méthodes capables de collecter, stocker, traiter et analyser des données à la fois volumineuses, rapides et hétérogènes. En sécurité réseau, ces données proviennent notamment de fichiers PCAP, flux réseau, journaux applicatifs, équipements de pare-feu, sondes IDS ou plateformes cloud.

1.4.1 Notion de Big Data et caractéristiques des 5V

Le Big Data se caractérise traditionnellement par cinq dimensions dites $5V$: volume, vélocité, variété, véracité et valeur [23, 25].

- Le **volume** fait référence à la quantité massive de données produites. Dans un réseau actif, le nombre de paquets et de flux peut rapidement devenir tel qu'une analyse manuelle est impossible, imposant des outils automatisés.
- La **vélocité** désigne la rapidité à laquelle les données sont générées et traitées. Le trafic réseau s'effectue en continu, parfois à très haute fréquence. Les systèmes de sécurité doivent traiter ces données suffisamment vite pour détecter une attaque avant qu'elle ne cause des dommages irréversibles.
- La **variété** correspond à la diversité des formats et sources : paquets bruts, flux réseau, journaux système, fichiers CSV, traces applicatives ou événements issus d'équipements hétérogènes.
- La **véracité** concerne la qualité et la fiabilité des données. Les traces réseau peuvent comporter des valeurs manquantes, infinies, aberrantes ou bruitées, nécessitant un prétraitement rigoureux pour garantir la fiabilité des données soumises à l'apprentissage.
- La **valeur** représente l'information pertinente extraite des données brutes. Ici, elle se traduit par la capacité des caractéristiques réseau à distinguer efficacement le trafic normal du trafic malveillant.

1.4.2 Hadoop, Spark et traitement distribué

Les plateformes de traitement distribué comme Apache Hadoop et Apache Spark ont été conçues pour relever les défis posés par le Big Data. Hadoop s'appuie sur un système de stockage distribué (*HDFS*) et un modèle de traitement par lots (*MapReduce*), permettant de répartir le traitement de grandes masses de données sur un cluster de nœuds. Spark complète ce paradigme en exploitant la mémoire distribuée pour accélérer considérablement les traitements, incluant également le traitement en flux continu (*streaming*) [24].

En cybersécurité, ces technologies facilitent l'analyse de volumes importants de journaux, flux réseau ou paquets en parallélisant les calculs et en traitant des ensembles de données trop vastes pour une machine unique.

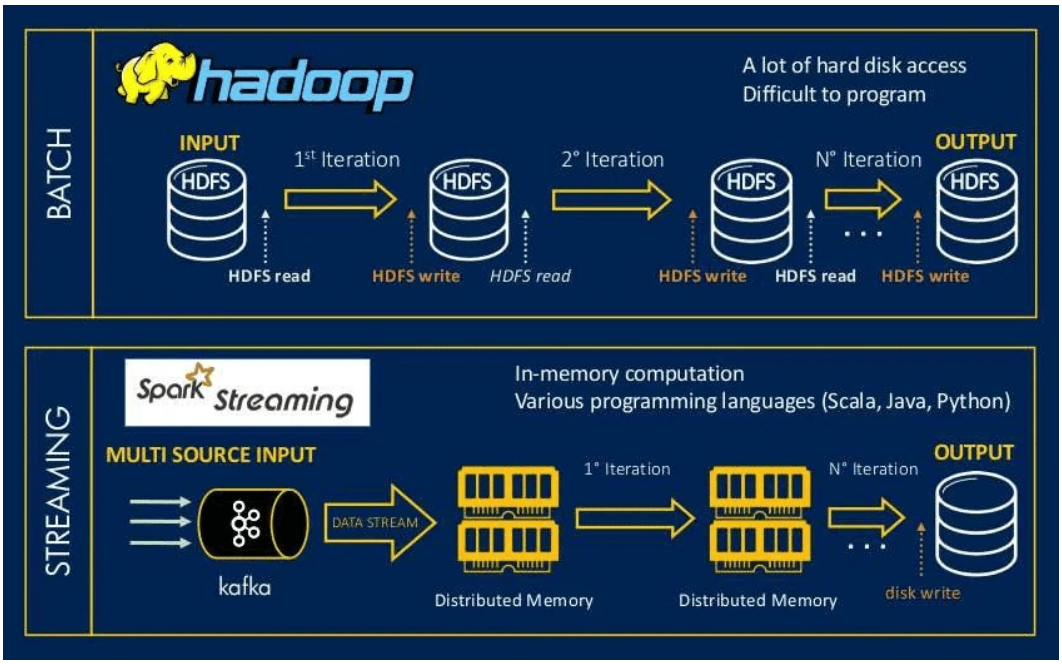

FIG. 1.5 : Comparaison du traitement par lots avec Hadoop et du traitement en mémoire avec Spark Streaming.

1.4.3 Pipeline Big Data pour l'analyse du trafic réseau

Un pipeline Big Data appliqué à la cybersécurité se compose généralement de quatre étapes successives :

- L'**acquisition** : recueille les données brutes : paquets, fichiers PCAP ou journaux.
- Le **nettoyage** : traite les valeurs manquantes, infinies ou incohérentes.
- La **transformation** : convertit les données brutes en caractéristiques exploitables.

- L'**analyse** : applique des méthodes statistiques ou des modèles d'apprentissage pour extraire l'information pertinente.

Dans le contexte du trafic réseau, ce pipeline se déploie concrètement comme suit : collecte des paquets, agrégation en flux, extraction des caractéristiques, nettoyage des données, normalisation des variables, puis entraînement des modèles de détection. Ce processus met en lumière que les performances d'un système de détection d'intrusions reposent autant sur la qualité des données et la rigueur du pipeline que sur le modèle final.

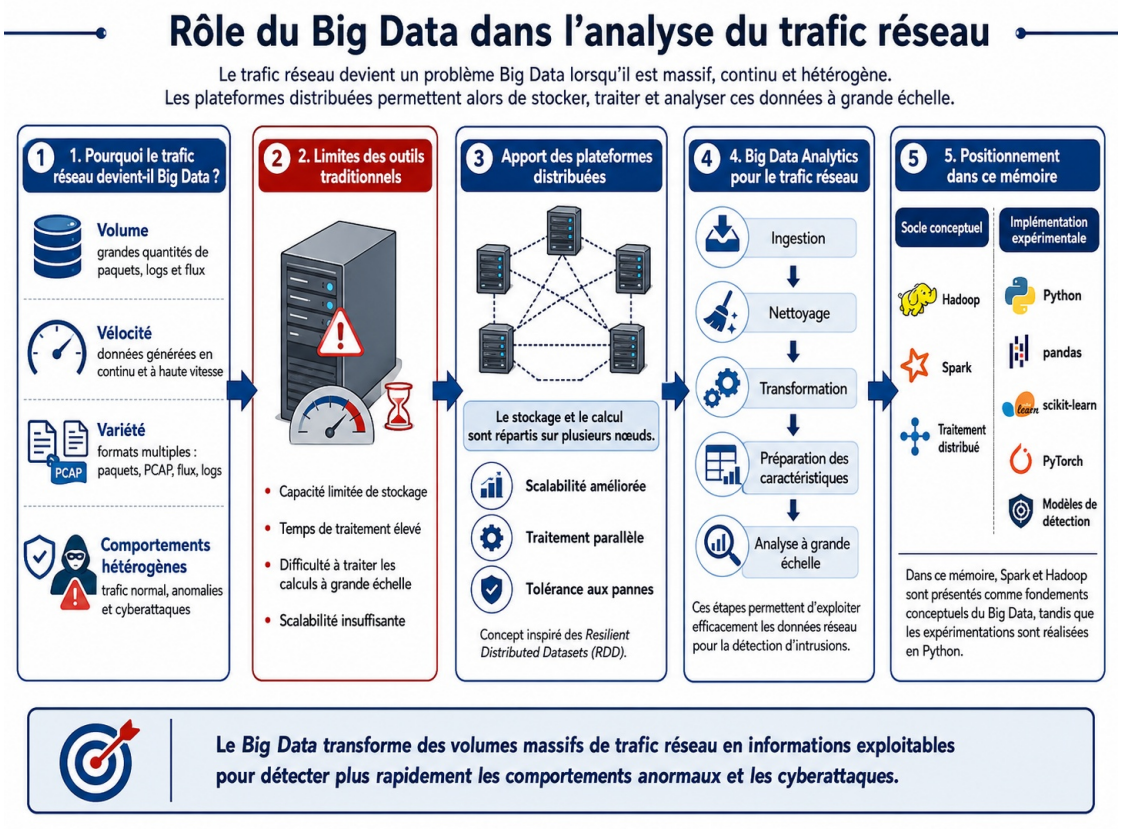

FIG. 1.6 : Rôle du Big Data dans l'analyse du trafic réseau et positionnement méthodologique de l'étude.

1.5 Jeu de données CICIDS2017

Le choix du jeu de données représente une étape cruciale dans toute étude de détection d'intrusions. Il doit fournir des exemples représentatifs du trafic normal et malveillant, contenir des caractéristiques exploitables, et permettre une évaluation rigoureuse des modèles. Pour cette recherche, le jeu de données **CICIDS2017** a été retenu comme référence expérimentale [19, 20].

Le dataset CICIDS2017 (*Canadian Institute for Cybersecurity Intrusion Detection Evaluation Dataset 2017*) constitue un standard dans la littérature sur la détection d'intrusions

réseau. Il présente des flux réseau enregistrés sur plusieurs jours, combinant du trafic légitime et diverses familles d'attaques. La forme tabulaire des données facilite leur exploitation directe par des algorithmes d'apprentissage automatique et profond [19].

Tab. 1.2 : Structure expérimentale des fichiers CICIDS2017 exploités dans les deux protocoles.

Fichier source	Rôle principal	Utilisation dans cette étude
Monday	Trafic normal majoritaire	Apprentissage du comportement BENIGN et base du protocole strict.
Tuesday	Attaques par force brute FTP/SSH	Entraînement strict et représentation des scénarios d'authentification.
Wednesday	Attaques DoS multiples	Validation et calibration des seuils dans le protocole strict.
Thursday	Attaques web et infiltration	Diversification des familles d'attaques dans le protocole global.
Friday	DDoS, PortScan et Botnet	Test strict sur Friday DDoS et contribution au protocole global.

1.5.1 Motivation expérimentale du choix de ce jeu de données

Plusieurs arguments justifient ce choix :

- Premièrement, CICIDS2017 offre une **large diversité de scénarios d'attaque** incluant DoS, DDoS, PortScan, Botnet, attaques web, force brute et infiltration, permettant d'évaluer les modèles sur un large spectre de comportements malveillants, proche des conditions réelles.
- Deuxièmement, il fournit des **étiquettes précises** (*labels*) décrivant la classe réelle de chaque flux, essentielles pour l'apprentissage supervisé et l'évaluation quantitative. La colonne *Label* constitue la vérité terrain (*ground truth*) de nos expérimentations.
- Troisièmement, les données sont déjà représentées sous forme de **flux caractérisés numériquement**, correspondant directement à la structure requise par nos modèles. Chaque observation décrit une communication par des propriétés statistiques mesurables.
- Quatrièmement, CICIDS2017 supporte plusieurs **protocoles expérimentaux** : une séparation stricte par journée pour évaluer la généralisation temporelle, ou un split aléatoire stratifié (75 % entraînement / 25 % test) pour une évaluation globale plus représentative.

1.5.2 Structure générale, scénarios et étiquetage

Le jeu est organisé en fichiers correspondant à différents jours et scénarios d'attaque. Certains contiennent majoritairement du trafic normal, d'autres ciblent des attaques spécifiques telles que DoS Hulk, DoS GoldenEye, DoS Slowloris, DDoS, PortScan, FTP-Patator, SSH-Patator, attaques web (XSS, SQL Injection, Heartbleed) et Botnet ARES.

Chaque flux est décrit par un ensemble de caractéristiques extraites des paquets composant la communication : durée, nombre de paquets, tailles, débit, temps inter-arrivée, *flags* TCP et autres indicateurs numériques. La colonne *Label* identifie chaque flux comme normal (*BENIGN*) ou associé à une catégorie d'attaque.

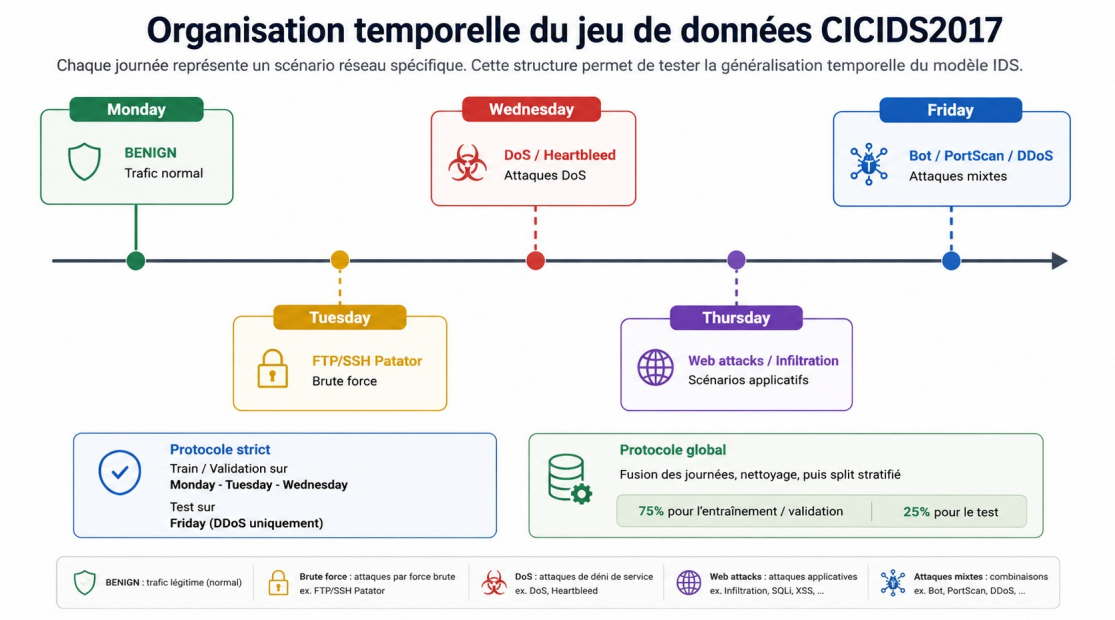

Fig. 1.7 : Organisation temporelle des journées et des scénarios d'attaque dans CICIDS2017, ainsi que leur rôle dans les deux protocoles expérimentaux.

Dans cette étude, l'étiquetage est simplifié en une **classification binaire** : *BENIGN* pour le trafic normal, *ATTACK* pour tout flux malveillant, quelle que soit sa nature. Cette simplification s'aligne avec le premier objectif d'un IDS, qui est de déterminer si un flux nécessite une alerte.

1.5.3 Prétraitement : nettoyage, normalisation et déséquilibre de classes

Avant l'entraînement, les données subissent un prétraitement rigoureux visant à améliorer leur qualité et éviter l'apprentissage sur des informations erronées ou parasites. Dans CICIDS2017, certaines colonnes comportent des valeurs manquantes (`NaN`), infinies (`Inf`) ou sans variance. Ces anomalies sont corrigées ou supprimées avant modélisation.

- Le **nettoyage** consiste à harmoniser les noms de colonnes, supprimer les doublons et traiter les valeurs `NaN` et `Inf`. Une **sélection de caractéristiques** élimine ensuite les variables constantes ou quasi-constantes, réduisant la dimensionnalité et améliorant la stabilité du modèle.
- La **normalisation** est essentielle pour compenser l'hétérogénéité des ordres de grandeur (durée, nombre de paquets, taille en octets). L'utilisation d'un `StandardScaler` centre et réduit les variables, limitant l'impact des valeurs extrêmes sur l'apprentissage.
- Le **déséquilibre des classes** (*class imbalance*) constitue un défi majeur. Certaines attaques sont sous-représentées par rapport au trafic normal, ce qui peut entraîner une prédominance de la classe majoritaire lors de l'apprentissage. L'évaluation doit donc s'appuyer sur des métriques adaptées telles que rappel (*recall*), précision (*precision*), F1-score et analyse de la matrice de confusion, au-delà de la seule précision globale (*accuracy*) [8, 9, 10].

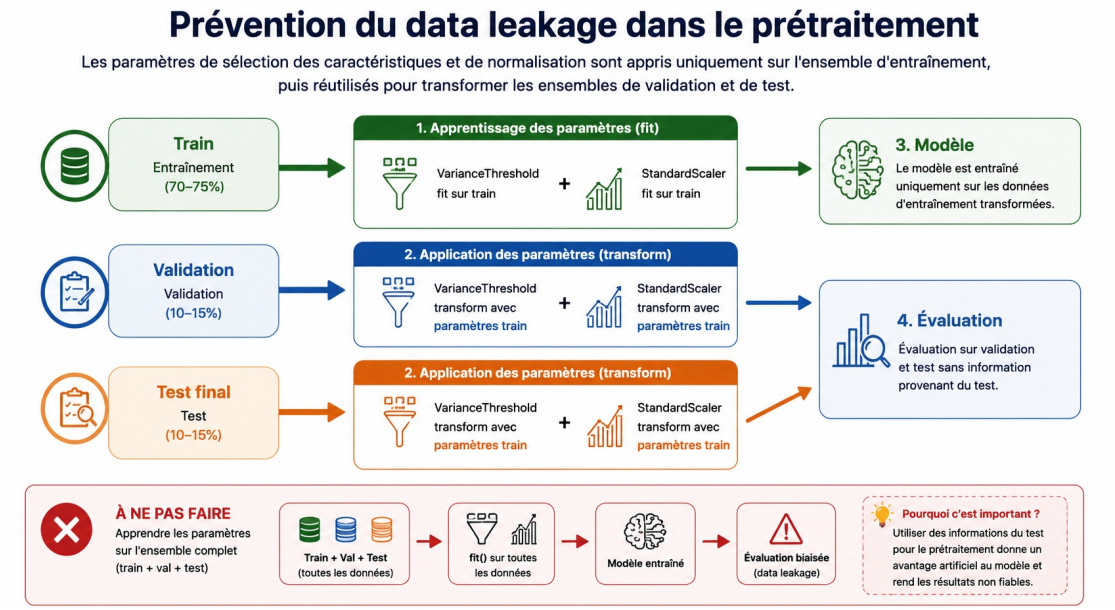

Fig. 1.8 : Prévention de la fuite de données lors du prétraitement : les transformations sont ajustées exclusivement sur l'ensemble d'entraînement, puis appliquées aux jeux de validation et test.

Une mauvaise gestion du prétraitement peut induire une **fuite de données** (*data leakage*), où des informations du jeu de test sont accidentellement utilisées lors de la formation, biaisant les performances. Pour éviter cela, toute opération apprise (sélection de caractéristiques, normalisation) est ajustée uniquement sur l'ensemble d'entraînement avant application aux données de validation et de test.

1.6 Conclusion

Ce chapitre a posé les bases conceptuelles et techniques indispensables à la compréhension du travail. Il a abordé les réseaux informatiques, leurs architectures et protocoles, la structuration des communications en paquets puis en flux, ainsi que la capture et l'extraction des caractéristiques via CICFlowMeter.

Les enjeux de la sécurité réseau ont été analysés sous l'angle du modèle CIA, de la surface d'attaque et des différentes familles d'attaques. Les limites des IDS/IPS traditionnels ont motivé l'adoption d'approches basées sur l'apprentissage automatique, exploitant les caractéristiques comportementales plutôt que les signatures seules.

La problématique du Big Data a été présentée à travers les cinq dimensions clés, les plateformes de traitement distribué et le pipeline d'analyse du trafic. Enfin, CICIDS2017 a été présenté comme support expérimental adapté, depuis sa structure jusqu'aux prétraitements requis.

Les éléments exposés ici constituent le socle pour les chapitres suivants, dont le chapitre 2 qui développera les fondements théoriques des modèles d'apprentissage profond retenus : MLP, CNN, autoencodeur et modèle hybride CNN-MLP, en explicitant leur pertinence pour la détection d'intrusions réseau.

Chapitre 2 : État de l'art et modèles d'apprentissage profond pour la détection d'intrusions

2.1 Introduction

Le chapitre précédent a présenté le contexte technique de ce travail : trafic réseau, paquets, flux, extraction de caractéristiques, systèmes IDS/IPS, jeu de données CICIDS2017 et principales opérations de prétraitement. Afin d'éviter toute répétition, ce chapitre ne revient pas sur ces notions. Il se concentre plutôt sur deux aspects : l'état de l'art des méthodes de détection d'intrusions fondées sur les données, puis les fondements mathématiques des modèles retenus. L'objectif de ce chapitre est donc double. D'une part, il s'agit de situer l'approche retenue par rapport aux approches existantes : méthodes traditionnelles, apprentissage automatique classique, apprentissage profond, autoencodeurs et modèles hybrides. D'autre part, il s'agit de justifier les choix méthodologiques de cette étude : l'utilisation d'un modèle hybride CNN-MLP supervisé et d'un Denoising Autoencoder comme approche complémentaire de détection d'anomalies.

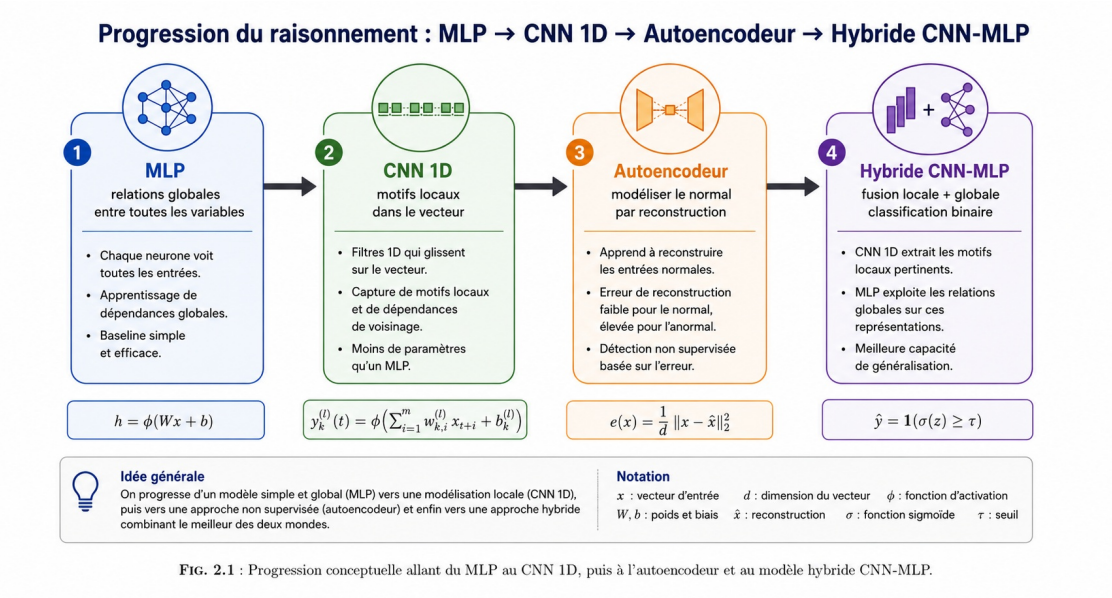

FIG. 2.1 : Progression conceptuelle allant du MLP au CNN 1D, puis à l'autoencodeur et au modèle hybride CNN-MLP.

FIG. 2.1 : Progression conceptuelle allant du MLP au CNN 1D, puis à l'autoencodeur et au modèle hybride CNN-MLP.

La figure 2.1 résume la progression logique suivie dans ce chapitre. Le MLP constitue la base naturelle pour des données tabulaires. Le CNN 1D introduit une lecture locale du vecteur de caractéristiques. L'autoencodeur apporte une logique de détection d'anomalies par reconstruction. Enfin, le modèle hybride CNN-MLP combine une représentation locale et une représentation globale du flux réseau.

2.2 État de l'art des approches de détection d'intrusions

2.2.1 Méthodes traditionnelles et transition vers l'apprentissage

Les systèmes de détection d'intrusions ont d'abord reposé sur des règles expertes et des signatures décrivant des comportements malveillants connus. Cette famille de méthodes reste pertinente dans de nombreux environnements opérationnels, car elle est interprétable et efficace lorsque l'attaque observée correspond à une signature déjà documentée [4, 5].

Cependant, ces approches montrent leurs limites lorsque l'attaque est nouvelle, modifiée, fragmentée ou dissimulée dans un trafic massif. Elles nécessitent également une mise à jour continue des bases de règles. Dans le contexte de cette étude, ces limites motivent le passage à des approches capables d'apprendre des régularités statistiques directement à partir des flux réseau.

L'intérêt de l'apprentissage automatique est précisément de remplacer une partie de l'expertise codée manuellement par un modèle entraîné sur des données. Au lieu de chercher une signature exacte, le modèle apprend des relations entre les caractéristiques du flux et la décision finale. Cette logique est mieux adaptée à des environnements où le trafic est volumineux, hétérogène et évolutif.

2.2.2 Apprentissage automatique classique pour les IDS

Les méthodes d'apprentissage automatique classique ont été largement étudiées pour la détection d'intrusions. Parmi les modèles les plus utilisés, on retrouve les arbres de décision, les forêts aléatoires, les machines à vecteurs de support, les k plus proches voisins, la régression logistique et les méthodes d'ensemble [8, 11].

Ces méthodes présentent plusieurs avantages. Elles sont relativement rapides à entraîner, efficaces sur des données tabulaires et parfois plus faciles à interpréter que les réseaux profonds. Les modèles fondés sur les arbres, en particulier, sont souvent performants lorsque les variables d'entrée sont déjà informatives.

Néanmoins, ces approches dépendent fortement de la qualité du prétraitement et de la représentation des données. Elles apprennent à partir de caractéristiques préconstruites et ne

produisent pas toujours des représentations internes aussi riches que celles des architectures profondes. De plus, des performances très élevées peuvent être obtenues artificiellement si le protocole expérimental n'est pas suffisamment contrôlé, par exemple lorsque des informations du test influencent la normalisation, la sélection de caractéristiques ou le choix du seuil.

L'approche retenue tient compte de cette difficulté en distinguant clairement l'entraînement, la validation et le test final. Les transformations dépendantes des données sont apprises sur l'entraînement uniquement, puis appliquées aux autres ensembles sans réajustement.

2.2.3 Apprentissage profond et détection d'intrusions

L'apprentissage profond permet de modéliser des relations non linéaires complexes entre les caractéristiques numériques extraites des flux réseau. Dans un système de détection d'intrusions, la décision ne dépend pas nécessairement d'une seule variable isolée, mais d'une combinaison de mesures comportementales telles que la durée du flux, le nombre de paquets, les volumes échangés, les temps inter-arrivée, les débits et les indicateurs TCP [13, 14].

Le perceptron multicouche (*Multi-Layer Perceptron*, MLP) constitue une architecture adaptée aux données tabulaires. Ses couches entièrement connectées apprennent des relations globales entre l'ensemble des caractéristiques du flux, sans imposer d'hypothèse particulière sur l'ordre des colonnes. Cette propriété est importante, car les variables discriminantes peuvent être éloignées dans le fichier CSV tout en contribuant conjointement à la distinction entre trafic normal et trafic malveillant.

Le réseau convolutif unidimensionnel (*CNN 1D*) applique des filtres sur le vecteur de caractéristiques afin d'extraire des interactions locales potentielles entre variables voisines. Cette utilisation doit rester prudente dans le cas de données tabulaires, car l'ordre des colonnes ne possède pas toujours une signification spatiale ou temporelle naturelle. Le CNN 1D fournit donc une lecture locale possible du vecteur, tandis que le MLP conserve une lecture globale de l'ensemble des variables.

Les autoencodeurs constituent une autre famille de modèles profonds orientée vers la détection d'anomalies. Leur principe consiste à apprendre la reconstruction de flux représentatifs du comportement normal, puis à utiliser l'erreur de reconstruction comme score d'anomalie [6, 16]. Une erreur élevée indique un écart important par rapport au comportement appris et peut conduire à classer le flux comme suspect. Cette approche dépend toutefois fortement du choix du seuil de décision et de la variabilité du trafic normal.

Le tableau 2.1 montre que le choix d'un modèle ne dépend pas uniquement de sa précision. Il dépend aussi du type de données disponible, de la présence ou non de labels

TAB. 2.1 : Comparaison synthétique des familles de travaux utilisées dans les IDS à base de données.

Famille	Données exploitées	Sortie	Atout principal	Limite principale
Signatures	Règles, motifs connus, journaux	Alerte binaire ou règle déclenchée	Interprétabilité élevée	Faible détection des attaques inconnues
ML classique	Caractéristiques de flux tabulaires	Classe ou score	Bon rapport performance/coût	Sensible au prétraitement et au choix des variables
Réseaux profonds	Vecteurs numériques, séquences ou flux	Probabilité de classe	Apprentissage de relations non linéaires	Besoin de données représentatives et de validation stricte
Autoencodeurs	Trafic normal ou majoritairement normal	Score d'anomalie	Adapté aux scénarios faiblement annotés	Seuil difficile à calibrer et faux négatifs possibles
Hybrides	Même observation lue par plusieurs branches	Probabilité ou score fusionné	Représentations complémentaires	Complexité plus élevée et interprétation plus délicate

fiables, du coût des fausses alertes et de la capacité du système à généraliser hors du scénario d'apprentissage.

2.2.4 Modèles hybrides et positionnement de l'approche retenue

Les modèles hybrides combinent plusieurs architectures afin d'exploiter des représentations complémentaires d'une même observation. Dans le cas des IDS, cette logique est pertinente lorsque les flux réseau sont décrits par des caractéristiques hétérogènes : certaines informations peuvent être liées à des groupes de variables proches, tandis que d'autres dépendent de relations globales entre l'ensemble des mesures.

L'architecture hybride CNN-MLP associe ainsi deux lectures du même vecteur de caractéristiques. La branche CNN 1D extrait une représentation locale potentielle, tandis que la branche MLP apprend une représentation globale. Ces deux sorties ne constituent pas deux décisions séparées : elles sont concaténées puis transmises à un classifieur final. Ce dernier produit un score d'attaque, transformé en probabilité par une fonction sigmoïde, puis comparé à un seuil de décision afin de classer le flux comme *BENIGN* ou *ATTACK*.

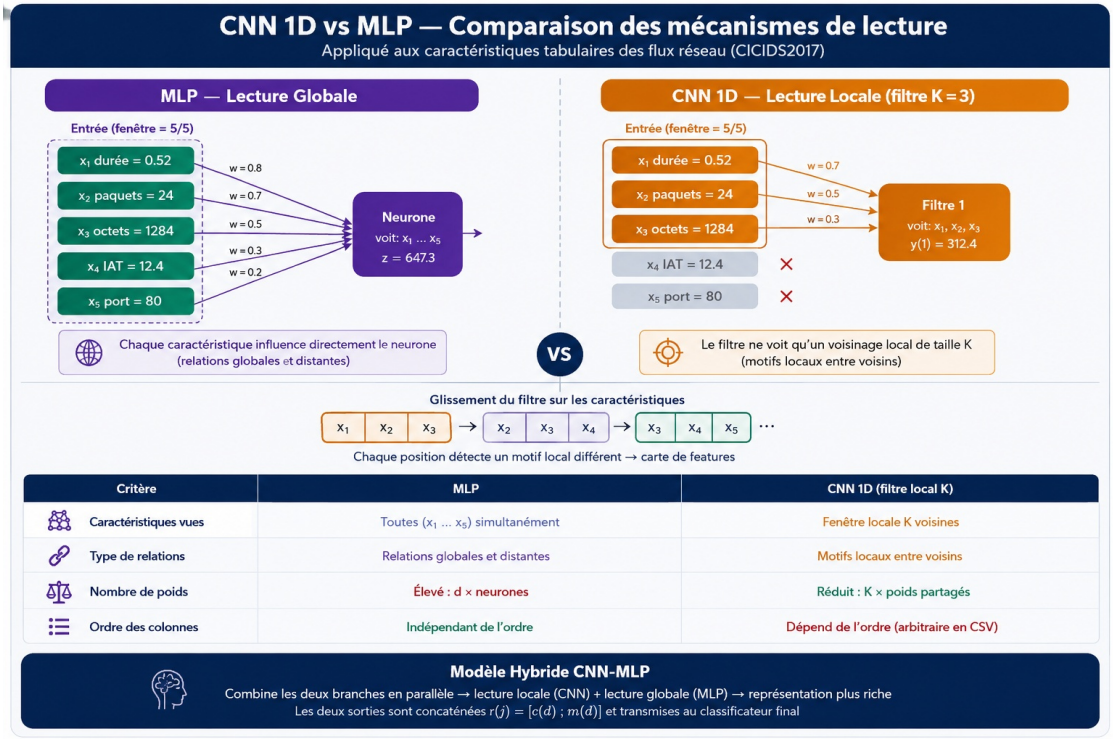

Fig. 2.2 : Comparaison entre la lecture globale du MLP et la lecture locale du CNN 1D dans une architecture hybride CNN-MLP.

La figure 2.2 illustre la complémentarité entre les deux branches de l'architecture hybride CNN-MLP. Le MLP réalise une lecture globale du vecteur de caractéristiques en considérant simultanément l'ensemble des variables du flux réseau. Cette lecture permet d'apprendre des relations générales, y compris entre des caractéristiques éloignées dans le tableau de données.

À l'inverse, le CNN 1D applique des filtres locaux sur des groupes restreints de caractéristiques voisines. Il peut ainsi extraire des motifs locaux potentiellement discriminants lorsque certaines variables proches portent une information commune. Dans l'architecture hybride, les sorties issues des deux branches sont concaténées afin de former une représentation plus riche du flux réseau. Cette représentation combinée est ensuite transmise au classifieur final chargé de produire la décision de classification.

Cette architecture se distingue d'un MLP seul, qui ne possède pas de mécanisme spécifique d'extraction locale. Elle se distingue également d'un CNN 1D seul, dont les performances peuvent dépendre de l'ordre des colonnes. Elle diffère enfin de l'autoencodeur, car elle appartient à un cadre supervisé et apprend directement une frontière de décision entre trafic normal et trafic malveillant.

Le tableau 2.2 met en évidence deux logiques complémentaires de détection. L'architecture hybride CNN-MLP relève d'une approche supervisée fondée sur les étiquettes *BENIGN* et *ATTACK*. Le Denoising Autoencoder relève d'une approche orientée anomalie, fondée sur l'écart entre un flux observé et le comportement normal appris.

Tab. 2.2 : Synthèse des approches de détection d'intrusions et positionnement de l'approche retenue.

Approche	**Principe**	**Avantages**	**Limites et positionnement**
Signatures et règles	Détection par motifs connus ou règles définies par des experts.	Interprétable et efficace sur les attaques déjà connues.	Faible adaptation aux variantes et attaques inconnues.
Machine learning classique	Classification à partir de caractéristiques numériques extraites du trafic.	Bon compromis entre performance, rapidité et simplicité.	Forte dépendance au prétraitement, à la sélection des variables et au protocole d'évaluation.
MLP	Réseau dense apprenant des relations globales entre toutes les variables.	Adapté aux données tabulaires et aux interactions non linéaires.	Absence de mécanisme explicite pour extraire des interactions locales.
CNN 1D	Filtres convolutifs appliqués au vecteur de caractéristiques.	Extraction d'interactions locales potentielles et partage de poids.	L'ordre des colonnes peut être arbitraire ; son usage doit donc être interprété avec prudence.
Autoencodeur	Apprentissage du comportement normal puis détection par erreur de reconstruction.	Intéressant pour la détection d'anomalies et les comportements atypiques.	Sensible au seuil de décision et à la variabilité du trafic normal.
Hybride CNN-MLP	Fusion d'une représentation locale potentielle et d'une représentation globale.	Exploite la complémentarité entre la lecture locale du CNN 1D et la lecture globale du MLP.	Architecture plus complexe ; l'apport de la branche CNN doit être vérifié expérimentalement.

2.3 Cadre général de l'apprentissage profond

L'apprentissage automatique désigne l'ensemble des méthodes permettant à un modèle d'améliorer ses décisions à partir de données. Dans le contexte étudié, il s'agit d'apprendre une fonction capable d'associer à un flux réseau une décision binaire : trafic normal ou attaque [11, 12].

L'apprentissage profond constitue une sous-famille de l'apprentissage automatique fondée sur des réseaux de neurones à plusieurs couches. Ces architectures permettent de modéliser des relations non linéaires complexes entre les variables d'entrée [13, 14]. Elles sont particulièrement adaptées lorsque le signal discriminant ne dépend pas d'une seule caractéristique, mais d'interactions entre plusieurs mesures.

Dans cette étude, chaque observation est représentée par un vecteur de caractéristiques numériques :

$$x = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d, \tag{2.1}$$

où d désigne le nombre de caractéristiques conservées après nettoyage et sélection. La tâche de détection est alors formulée comme l'apprentissage d'une fonction de décision :

$$f_\theta : \mathbb{R}^d \to \{0, 1\}, \tag{2.2}$$

où 0 désigne un flux normal et 1 un flux d'attaque.

Deux paradigmes sont considérés. Dans l'apprentissage supervisé, le modèle apprend à partir de couples (x_i, y_i) où l'étiquette y_i est connue. Dans l'apprentissage non supervisé ou semi-supervisé orienté anomalie, le modèle apprend principalement la structure du trafic normal et signale les observations qui s'en écartent fortement.

La figure 2.3 permet de situer les deux familles de modèles retenues. Le modèle hybride CNN-MLP relève de la classification supervisée, tandis que le Denoising Autoencoder relève d'une logique de détection d'anomalies par reconstruction.

2.4 Perceptron multicouche (MLP)

2.4.1 Architecture et propagation avant

Le perceptron multicouche, ou *Multi-Layer Perceptron* (MLP), est un réseau de neurones composé d'une couche d'entrée, de plusieurs couches cachées entièrement connectées et d'une couche de sortie. Chaque neurone calcule une combinaison linéaire de ses entrées, ajoute un biais, puis applique une fonction d'activation non linéaire [13].

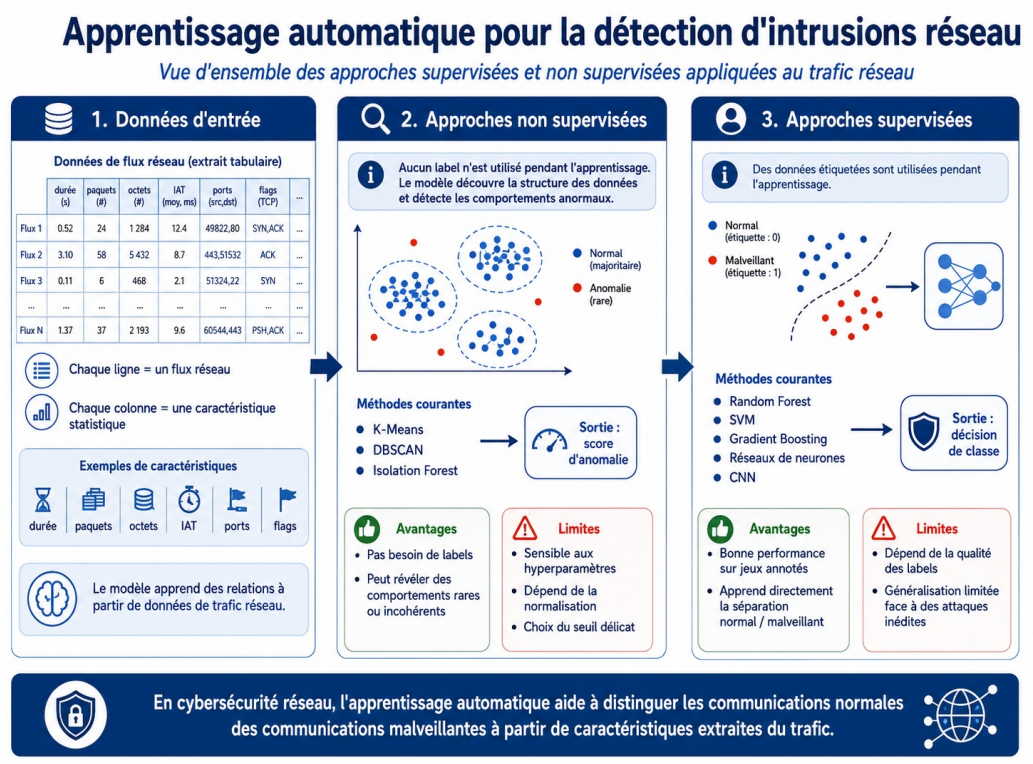

FIG. 2.3 : Positionnement des approches supervisées et orientées anomalie pour la détection d'intrusions.

Pour la couche l, la propagation avant s'écrit :

$$z^{(l)} = W^{(l)} h^{(l-1)} + b^{(l)}, \qquad h^{(l)} = \phi\big(z^{(l)}\big), \tag{2.3}$$

où $W^{(l)}$ est la matrice de poids, $b^{(l)}$ le vecteur de biais et ϕ la fonction d'activation. L'entrée du réseau est notée $h^{(0)} = x$.

La fonction d'activation utilisée dans les couches cachées est généralement ReLU :

$$\mathrm{ReLU}(u) = \max(0, u). \tag{2.4}$$

Elle facilite l'apprentissage de relations non linéaires et réduit certains problèmes liés au gradient évanescent.

Pour une classification binaire, la couche finale produit un logit z, transformé en probabilité par la fonction sigmoïde :

$$p(y = 1 \mid x) = \sigma(z) = \frac{1}{1 + e^{-z}}. \tag{2.5}$$

La décision finale est obtenue par comparaison à un seuil τ :

$$\hat{y} = \mathbb{1}[\sigma(z) \geq \tau] \,. \tag{2.6}$$

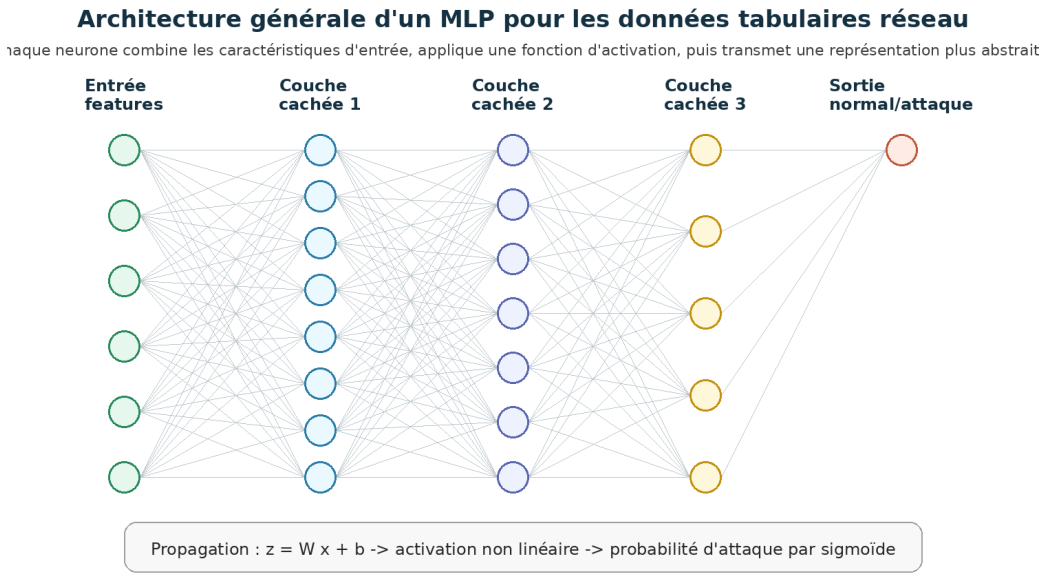

FIG. 2.4 : Architecture générale d'un perceptron multicouche appliqué aux caractéristiques tabulaires du trafic réseau.

La figure 2.4 montre que le MLP traite l'ensemble du vecteur d'entrée de manière globale. Cette propriété est utile pour les données tabulaires, car une attaque peut être révélée par une combinaison de caractéristiques éloignées dans le fichier CSV.

2.4.2 Régularisation du MLP

L'entraînement d'un MLP sur des données de trafic peut conduire au surapprentissage, notamment lorsque le modèle devient trop spécialisé sur l'ensemble d'entraînement. Deux techniques sont particulièrement utiles pour limiter ce problème.

La normalisation par lot, ou *Batch Normalization*, stabilise les activations intermédiaires en les centrant et en les réduisant à l'échelle du mini-lot :

$$\hat{h}^{(l)} = \frac{h^{(l)} - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \varepsilon}}, \qquad \tilde{h}^{(l)} = \gamma \hat{h}^{(l)} + \beta. \tag{2.7}$$

Cette opération accélère la convergence et réduit la sensibilité à l'initialisation [17].

Le dropout désactive aléatoirement une fraction des neurones pendant l'entraînement :

$$\tilde{h}_j^{(l)} = \begin{cases} h_j^{(l)}/(1-p) & \text{avec probabilité } 1-p, \\ 0 & \text{avec probabilité } p. \end{cases} \tag{2.8}$$

Il force le réseau à ne pas dépendre d'un petit nombre de neurones et améliore ainsi sa capacité de généralisation [18].

2.4.3 Rôle du MLP dans l'approche retenue

Dans l'étude, le MLP n'est pas retenu comme modèle final isolé, mais comme composant d'une architecture hybride. Son rôle est d'apprendre des relations globales entre toutes les caractéristiques du flux. Par exemple, une durée courte, un volume élevé, un port particulier et certains indicateurs TCP peuvent, ensemble, constituer un motif discriminant.

Cette lecture globale complète la lecture locale fournie par la branche CNN 1D. Le MLP permet donc d'éviter que le modèle ne dépende uniquement de voisinages de colonnes dont l'ordre peut être arbitraire.

2.5 Réseaux convolutifs 1D (CNN 1D)

2.5.1 Principe de la convolution 1D

Les réseaux de neurones convolutifs ont été initialement développés pour les images, où les filtres détectent des motifs locaux dans une grille de pixels [13, 15]. Dans une version unidimensionnelle, le même principe est appliqué à une séquence ou à un vecteur. Un filtre de taille K glisse le long du vecteur d'entrée afin de produire une carte de caractéristiques.

Pour un filtre k, la sortie à la position t peut être écrite :

$$y_k(t) = \phi\left(b_k + \sum_{i=0}^{K-1} w_{k,i} x_{t+i}\right), \tag{2.9}$$

où $w_{k,i}$ sont les poids du filtre, b_k le biais et ϕ la fonction d'activation. En appliquant plusieurs filtres, le réseau apprend différentes familles de motifs locaux.

La figure 2.5 illustre l'utilisation d'un CNN 1D sur un vecteur de caractéristiques. Le modèle ne traite pas une image, mais une suite de variables numériques. Les filtres convolutifs cherchent des combinaisons locales entre caractéristiques voisines, puis les couches denses exploitent les représentations obtenues pour produire une décision.

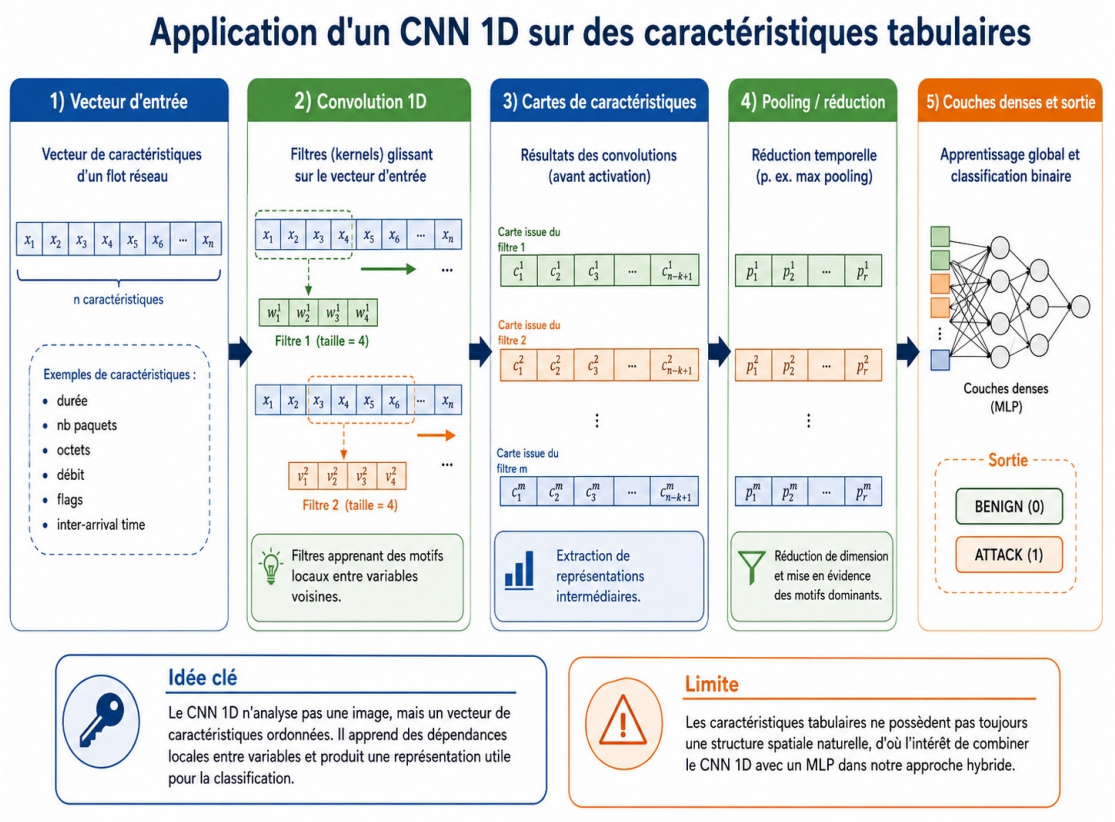

FIG. 2.5 : Application d'une convolution 1D sur un vecteur de caractéristiques tabulaires issues d'un flux réseau.

2.5.2 Intérêt et limites sur données tabulaires

L'intérêt du CNN 1D réside dans sa capacité à apprendre des motifs locaux avec un nombre limité de paramètres. Sur des données de trafic, certains groupes de variables peuvent décrire des comportements cohérents, comme les tailles de paquets, les temps inter-arrivée ou certains indicateurs liés aux échanges TCP.

Cependant, l'utilisation d'un CNN 1D sur des données tabulaires doit rester prudente. Dans une image, deux pixels voisins ont une proximité spatiale réelle. Dans un fichier CSV, deux colonnes voisines ne sont pas nécessairement liées. L'ordre des variables peut dépendre de l'outil d'extraction ou du choix de prétraitement. Ainsi, un CNN 1D utilisé seul pourrait accorder trop d'importance à des voisinages artificiels.

Pour cette raison, l'approche retenue n'utilise pas le CNN 1D comme modèle unique. Il est employé comme branche locale dans une architecture hybride, où la branche MLP conserve une lecture globale de toutes les caractéristiques.

2.6 Denoising Autoencoder pour la détection d'anomalies

2.6.1 Principe de l'autoencodeur

Un autoencodeur est un réseau de neurones entraîné à reconstruire son entrée. Il est composé d'un encodeur f_θ, qui projette l'entrée dans un espace latent, et d'un décodeur g_φ, qui reconstruit l'entrée à partir de cette représentation :

$$z = f_\theta(x), \qquad \hat{x} = g_\varphi(z). \tag{2.10}$$

L'apprentissage minimise l'erreur de reconstruction :

$$\mathcal{L}_{AE} = \frac{1}{n} \sum_{i=1}^{n} \|x_i - g_\varphi(f_\theta(x_i))\|_2^2 . \tag{2.11}$$

L'intérêt de cette approche pour les IDS est le suivant : si l'autoencodeur est entraîné principalement sur du trafic normal, il apprend à reconstruire correctement ce type de comportement. Un flux anormal, différent de la structure apprise, sera moins bien reconstruit et produira une erreur plus élevée [6, 7].

2.6.2 Autoencodeur débruiteur

Le Denoising Autoencoder ajoute volontairement du bruit à l'entrée pendant l'entraînement :

$$\tilde{x} = x + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \sigma^2). \tag{2.12}$$

Le modèle reçoit l'entrée bruitée $\tilde{x}$, mais doit reconstruire l'entrée propre x. Sa fonction de perte devient :

$$\mathcal{L}_{DAE} = \frac{1}{n} \sum_{i=1}^{n} \|x_i - g_\varphi(f_\theta(\tilde{x}_i))\|_2^2 . \tag{2.13}$$

Cette stratégie empêche le réseau d'apprendre une simple copie de l'entrée. Elle l'oblige à extraire une structure plus robuste du trafic normal [16].

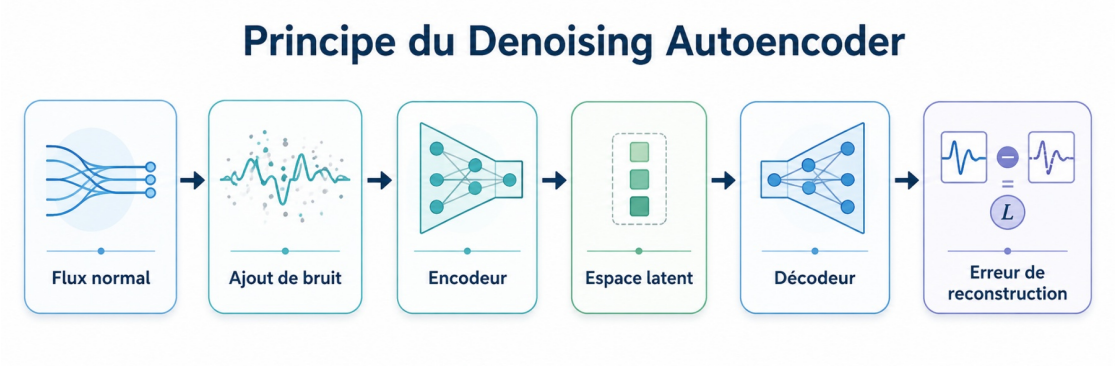

FIG. 2.6 : Principe mathématique et décisionnel du Denoising Autoencoder.

La figure 2.6 montre que le Denoising Autoencoder suit une logique différente du classifieur supervisé. Il ne cherche pas directement une frontière entre deux classes pendant l'entraînement ; il apprend d'abord à reconstruire le comportement normal, puis utilise l'erreur de reconstruction comme signal d'anomalie.

2.6.3 Score d'anomalie et seuil de décision

À l'inférence, le score d'anomalie d'un flux est défini par son erreur quadratique moyenne de reconstruction :

$$e(x) = \frac{1}{d} \sum_{j=1}^{d} (x_j - \hat{x}_j)^2 . \tag{2.14}$$

La décision finale est obtenue par comparaison avec un seuil τ_{AE} :

$$\hat{y} = \begin{cases} 1 \text{ (ATTACK)} & \text{si } e(x) \geq \tau_{AE}, \\ 0 \text{ (BENIGN)} & \text{si } e(x) < \tau_{AE}. \end{cases} \tag{2.15}$$

Le seuil joue un rôle central. Un seuil trop faible augmente les faux positifs, tandis qu'un seuil trop élevé augmente les faux négatifs. Dans la démarche adoptée, le seuil est calibré sur l'ensemble de validation et n'est jamais choisi sur le test final.

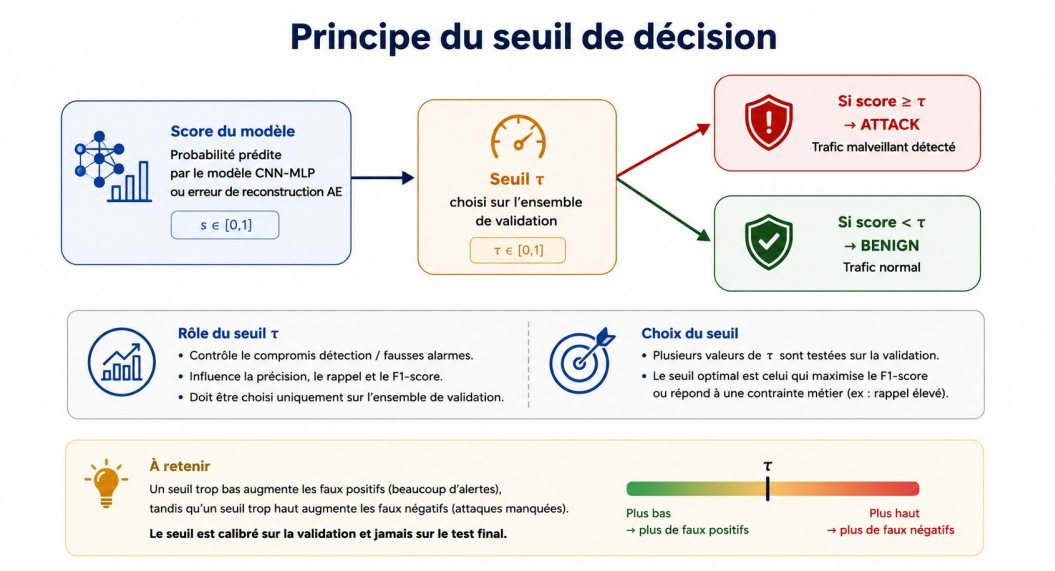

FIG. 2.7 : Principe du seuil de décision transformant un score continu en décision binaire.

La figure 2.7 illustre le rôle du seuil dans la transformation d'un score continu en décision IDS. Cette étape est commune aux deux modèles étudiés : le modèle hybride produit une probabilité d'attaque, tandis que l'autoencodeur produit une erreur de reconstruction.

2.7 Modèle hybride CNN-MLP

2.7.1 Motivation

Les sections précédentes montrent que le MLP et le CNN 1D possèdent des qualités complémentaires. Le MLP apprend des relations globales entre toutes les variables, mais il ne dispose pas d'un mécanisme spécifique pour extraire des motifs locaux. Le CNN 1D apprend des combinaisons locales, mais il peut être limité par l'ordre arbitraire des colonnes.

Le modèle hybride CNN-MLP vise à combiner ces deux lectures. Le même vecteur de caractéristiques est transmis à deux branches parallèles : une branche CNN 1D et une branche MLP. Les représentations obtenues sont ensuite fusionnées pour alimenter un classifieur final.

2.7.2 Architecture

La figure 2.8 présente le principe de l'architecture proposée. La branche CNN 1D extrait des motifs locaux, tandis que la branche MLP apprend des relations globales. La concaténa-

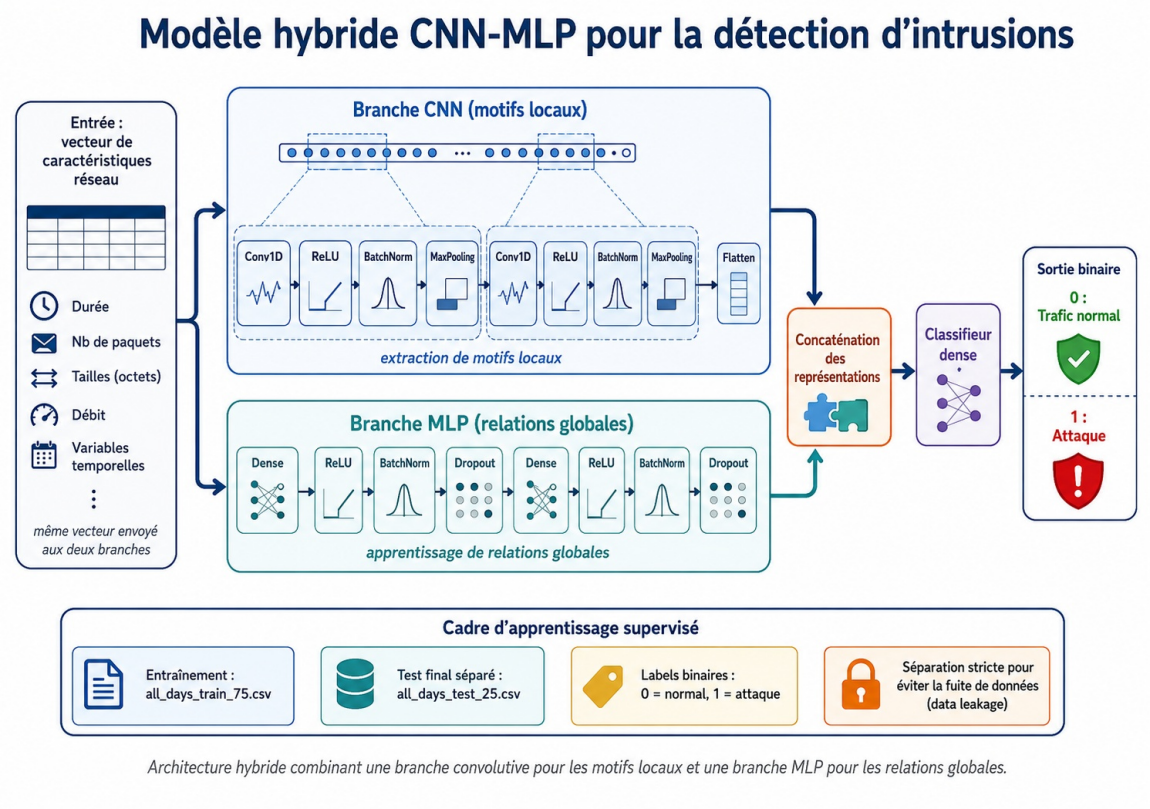

FIG. 2.8 : Architecture du modèle hybride CNN-MLP combinant lecture locale et lecture globale du flux.

tion des deux représentations donne au classifieur final une information plus complète sur le flux.

Soit $x \in \mathbb{R}^d$ le vecteur d'entrée. La branche CNN produit une représentation locale :

$$c(x) = \text{Pool}(\text{Conv1D}(x; W_c, b_c)) \in \mathbb{R}^{d_c}. \tag{2.16}$$

La branche MLP produit une représentation globale :

$$m(x) = h_{\text{MLP}}^{(L)}(x) \in \mathbb{R}^{d_m}. \tag{2.17}$$

Les deux représentations sont concaténées :

$$r(x) = [c(x); m(x)] \in \mathbb{R}^{d_c+d_m}. \tag{2.18}$$

Le classifieur final produit ensuite un logit z, transformé en probabilité d'attaque par une sigmoïde :

$$z = W_r r(x) + b_r, \qquad p(y = 1 \mid x) = \sigma(z). \tag{2.19}$$

La décision finale est :

$$\hat{y} = \mathbb{1}[\sigma(z) \geq \tau]\,. \tag{2.20}$$

2.7.3 Fonction de perte pondérée

Le modèle hybride est entraîné par entropie croisée binaire. Lorsque la classe attaque est moins représentée que la classe normale, il est nécessaire de pondérer la perte afin d'éviter que le modèle ne favorise excessivement la classe majoritaire [8].

Le poids associé à la classe positive est défini par :

$$\text{pos\_weight} = \frac{N_{\text{normal}}}{N_{\text{attack}}}. \tag{2.21}$$

La perte pondérée s'écrit :

$$\mathcal{L}_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}\left[w y_i \log(\sigma(z_i)) + (1-y_i)\log(1-\sigma(z_i))\right], \tag{2.22}$$

où $w = \text{pos\_weight}$. Cette pondération augmente l'importance des erreurs sur la classe attaque, ce qui est cohérent avec les contraintes d'un IDS.

2.7.4 Différence avec les approches existantes

La différence principale de l'approche retenue réside dans la comparaison explicite entre un modèle supervisé hybride et un modèle orienté anomalie. Le modèle hybride CNN-MLP exploite les étiquettes pour apprendre une frontière de décision entre BENIGN et ATTACK. Le Denoising Autoencoder, lui, apprend la structure du comportement normal et utilise l'erreur de reconstruction pour signaler les flux atypiques.

Cette comparaison permet de répondre à une question pratique : dans un contexte IDS, vaut-il mieux privilégier un classifieur supervisé performant sur des attaques connues, ou conserver une approche d'anomalie plus générale mais potentiellement plus sensible aux faux positifs ? Les résultats du chapitre 4 permettront de discuter cette question expérimentalement.

2.8 Métriques d'évaluation IDS

2.8.1 Matrice de confusion

L'évaluation d'un IDS ne doit pas se limiter à l'exactitude globale (*accuracy*). Dans un jeu de données déséquilibré, un modèle peut obtenir une accuracy élevée tout en détectant mal la classe attaque. Pour cette raison, l'analyse doit partir de la matrice de confusion [9].

Dans le cas binaire étudié, les quatre décisions possibles sont :

- **Vrai positif (VP)** : flux d'attaque correctement détecté.
- **Faux négatif (FN)** : flux d'attaque classé comme normal.
- **Faux positif (FP)** : flux normal classé comme attaque.
- **Vrai négatif (VN)** : flux normal correctement classé.

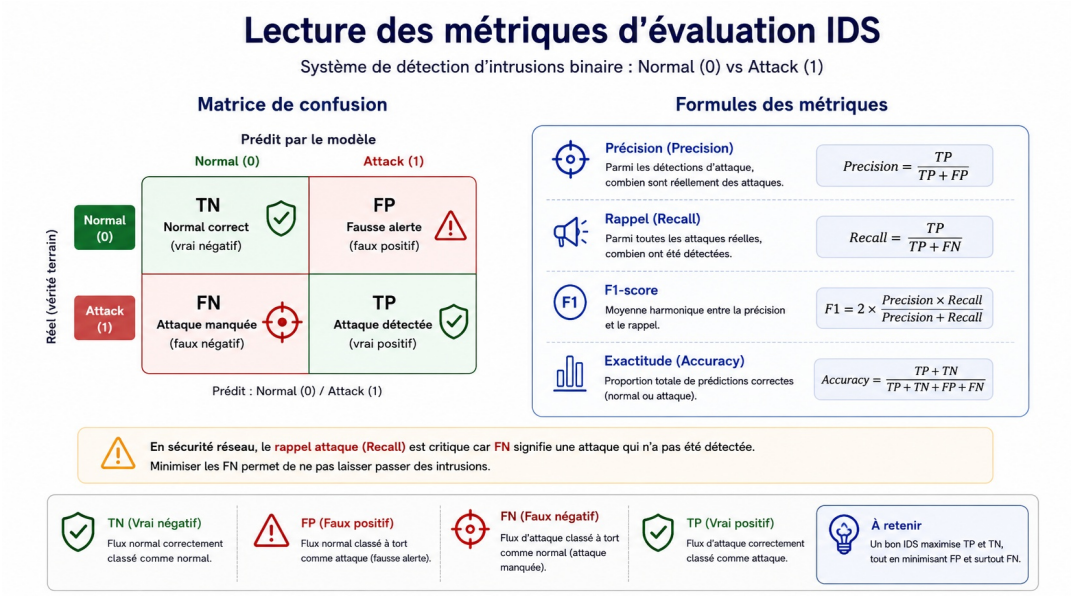

FIG. 2.9 : Matrice de confusion et formules des métriques d'évaluation pour un IDS binaire.

La figure 2.9 met en évidence le rôle critique des faux négatifs en sécurité réseau. Un faux positif produit une alerte inutile, mais un faux négatif correspond à une attaque non détectée.

2.8.2 Précision, rappel, F1-score et balanced accuracy

La précision mesure la fiabilité des alertes émises :

$$\text{Précision} = \frac{VP}{VP + FP}. \tag{2.23}$$

Le rappel mesure la proportion d'attaques correctement détectées :

$$\text{Rappel} = \frac{VP}{VP + FN}. \tag{2.24}$$

Le F1-score combine la précision et le rappel :

$$\text{F1-score} = 2 \cdot \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}. \tag{2.25}$$

L'exactitude globale est définie par :

$$\text{Accuracy} = \frac{VP + VN}{VP + VN + FP + FN}. \tag{2.26}$$

La balanced accuracy est particulièrement utile lorsque les classes sont déséquilibrées :

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{VP}{VP + FN} + \frac{VN}{VN + FP}\right). \tag{2.27}$$

Dans un IDS, le rappel de la classe attaque est souvent prioritaire, car il mesure la capacité du système à ne pas laisser passer une intrusion. La précision reste également importante, car un grand nombre de faux positifs peut saturer les analystes et réduire la confiance dans le système. Le F1-score offre donc un compromis utile entre détection et qualité des alertes.

2.9 Synthèse du positionnement méthodologique

À partir de l'état de l'art présenté dans ce chapitre, le positionnement méthodologique peut être résumé autour de quatre choix principaux :

- L'analyse est conduite au niveau des caractéristiques de flux plutôt qu'au niveau du contenu applicatif. Ce choix est cohérent avec les contraintes de confidentialité et avec la présence fréquente de trafic chiffré.
- L'architecture retenue utilise un modèle hybride CNN-MLP afin de combiner deux lectures du même flux : une lecture locale par convolution 1D et une lecture globale par MLP. Ce choix vise à dépasser les limites d'un MLP seul et d'un CNN 1D utilisé isolément.
- L'évaluation inclut également un Denoising Autoencoder afin de disposer d'une approche orientée anomalie. Cette comparaison permet d'étudier non seulement la performance supervisée, mais aussi l'intérêt d'un modèle qui apprend principalement la structure du trafic normal.

- L'évaluation est conduite avec des métriques adaptées aux IDS : précision, rappel, F1-score, balanced accuracy, faux positifs et faux négatifs. Cette lecture est plus pertinente que l'accuracy seule, notamment en présence de classes déséquilibrées.

2.10 Conclusion

Ce chapitre a présenté l'état de l'art et les fondements théoriques nécessaires à la compréhension des modèles utilisés. Les méthodes traditionnelles à signatures restent utiles contre les attaques connues, mais elles sont limitées face aux comportements nouveaux et aux variantes d'attaques. Les méthodes classiques d'apprentissage automatique améliorent cette situation en apprenant à partir de caractéristiques de flux, mais elles restent fortement dépendantes du prétraitement et du protocole d'évaluation.

Les modèles profonds apportent une capacité supplémentaire de représentation. Le MLP apprend des relations globales entre variables, le CNN 1D extrait des motifs locaux, l'autoencodeur détecte les anomalies par reconstruction, et le modèle hybride CNN-MLP fusionne deux lectures complémentaires du flux.

Le positionnement retenu repose sur la comparaison contrôlée de deux philosophies de détection : un modèle supervisé hybride CNN-MLP et un Denoising Autoencoder orienté anomalie. Le chapitre suivant décrit la méthodologie expérimentale, l'organisation des scripts, la préparation des données et les protocoles d'entraînement utilisés pour évaluer ces deux modèles.

Chapitre 3 : Environnement, méthodologie et implémentation

3.1 Introduction du chapitre

Les chapitres précédents ont présenté le contexte de la détection d'intrusions réseau, les caractéristiques du trafic TCP/UDP, ainsi que les deux familles de modèles retenues pour cette étude : le modèle hybride CNN-MLP et le Denoising Autoencoder. Le présent chapitre ne se limite pas à décrire l'exécution des scripts. Il explique, étape par étape, comment chaque partie du pipeline a été construite, pourquoi elle est nécessaire d'un point de vue mathématique, et comment elle est traduite algorithmiquement dans le code.

L'objectif est donc de relier systématiquement trois niveaux complémentaires :

1. le **niveau méthodologique**, qui décrit le rôle de l'étape dans le système IDS ;
2. le **niveau mathématique et algorithmique**, qui justifie le traitement appliqué ;
3. le **niveau implémentation**, qui montre uniquement les fragments de code essentiels, sans insérer les scripts complets dans le manuscrit.

Cette organisation évite ainsi une présentation purement descriptive du code. Chaque fragment est introduit comme la conséquence directe d'un choix méthodologique ou mathématique. Les fragments présentés ont un rôle explicatif : ils décrivent la logique essentielle du pipeline sans transformer le manuscrit en documentation de programmation complète.

Pendant la mise en place du pipeline, une petite erreur dans le choix des colonnes suffisait à rendre les scores artificiellement élevés. Les colonnes `Label`, `LabelBinary`, `AttackType` et `SourceFile` ont donc été traitées comme des informations de suivi et jamais comme des variables d'entrée. Cette décision simple a guidé toute l'implémentation.

Le chapitre est structuré autour de deux protocoles expérimentaux. Le premier protocole, dit *strict*, cherche à mesurer la robustesse temporelle des modèles : l'entraînement est réalisé sur certains jours du dataset CICIDS2017, tandis que le test final est effectué sur un jour séparé contenant l'attaque Friday DDoS. Le second protocole, dit *global*, fusionne les huit fichiers CSV du dataset, applique un split stratifié 75 %/25 % et mesure la performance sur une diversité plus large de familles d'attaques.

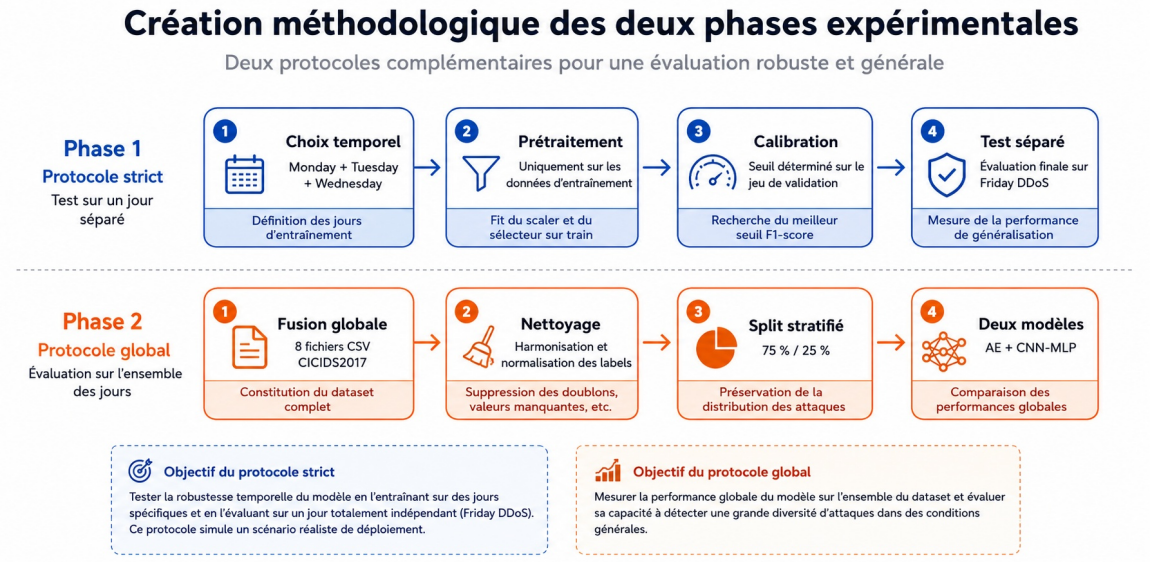

Fig. 3.1 : Vue générale des étapes de création des deux phases expérimentales.

La figure 3.1 résume la logique suivie dans ce chapitre. Les données brutes ne sont jamais envoyées directement vers les modèles. Elles passent d'abord par des opérations de nettoyage, de sélection de variables, de normalisation et de séparation expérimentale. Ce principe est important, car un modèle d'apprentissage profond n'est fiable que si les données d'entrée sont cohérentes, numériques et préparées sans fuite d'information entre l'entraînement et le test.

3.2 Environnement matériel et logiciel

Les expérimentations ont été réalisées dans un environnement Python isolé afin de garantir la reproductibilité des dépendances. L'utilisation de PyTorch permet de construire les architectures neuronales, tandis que scikit-learn assure les opérations classiques de prétraitement, de split stratifié et d'évaluation.

Tab. 3.1 : Environnement matériel utilisé pour les expérimentations.

Élément	Spécification
Ordinateur	Lenovo Legion 5 Pro
Processeur	AMD Ryzen 7 5800H, 3.20 GHz
Mémoire vive	16 GB RAM
Carte graphique	NVIDIA GeForce RTX 3070 Laptop GPU, 8 GB
Système d'exploitation	Windows 11, 64 bits

Tab. 3.2 : Outils logiciels et rôle dans le pipeline.

Outil	Rôle
Python	Langage principal d'implémentation
PyTorch	Construction et entraînement du CNN-MLP et de l'Autoencoder
CUDA	Accélération GPU des calculs matriciels
pandas	Chargement, nettoyage et fusion des fichiers CSV
NumPy	Manipulation numérique des matrices et vecteurs
scikit-learn	Split stratifié, sélection de variables, normalisation et métriques
Matplotlib	Génération des courbes et figures de résultats
joblib	Sauvegarde des objets de prétraitement ajustés

Le tableau suivant résume les principaux choix expérimentaux. Les valeurs exactes peuvent être légèrement ajustées selon le script exécuté, mais elles décrivent la configuration conservée pour produire les résultats discutés dans le chapitre 4.

Tab. 3.3 : Hyperparamètres principaux des modèles expérimentaux.

Paramètre	Hybrid CNN-MLP	Denoising Autoencoder
Optimiseur	Adam	Adam
Fonction de perte	BCEWithLogitsLoss pondérée	MSE de reconstruction
Batch size	1024	1024
Learning rate	10^{-3}	10^{-3}
Époques maximales	30 à 50 selon la phase	30 à 50 selon la phase
Patience	Early stopping sur validation	Early stopping sur validation
Pondération de classe	Calculée à partir du rapport BENIGN/ATTACK du train	Non applicable
Dimension latente	Non applicable	Représentation comprimée de l'entrée
Bruit d'entrée	Non utilisé	Bruit gaussien pendant l'apprentissage
Seuil final	Calibré sur validation	Calibré par percentiles de l'erreur
Seed	Fixée pour NumPy, PyTorch et split	Fixée pour NumPy, PyTorch et split

L'utilisation du GPU est sélectionnée automatiquement lorsque CUDA est disponible. Ce choix n'a pas d'impact sur la définition mathématique des modèles, mais il réduit fortement le temps d'entraînement, car les multiplications matricielles sont exécutées en parallèle.

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)
```

Listing 3.1: Sélection automatique du support de calcul.

D'un point de vue algorithmique, cette instruction signifie que toutes les opérations de propagation avant, de calcul de perte et de rétropropagation seront exécutées sur le même support de calcul. Le modèle et les tenseurs doivent être placés sur le même `device`, sinon PyTorch produit une erreur de cohérence.

3.3 Organisation générale du pipeline expérimental

Le pipeline global peut être vu comme une succession de transformations appliquées à un ensemble de flux réseau. Un flux est représenté par un vecteur numérique :

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{id}) \in \mathbb{R}^d, \tag{3.1}$$

où d désigne le nombre de caractéristiques retenues après nettoyage et sélection. Chaque flux possède également une étiquette binaire :

$$y_i = \begin{cases} 0, & \text{si le flux est BENIGN,} \\ 1, & \text{si le flux est une attaque.} \end{cases} \tag{3.2}$$

Le pipeline expérimental se compose de cinq blocs principaux : chargement des données, nettoyage, séparation expérimentale, prétraitement numérique, entraînement et décision. La différence entre les deux phases ne concerne pas uniquement les fichiers utilisés ; elle concerne aussi l'objectif de validation. La phase stricte évalue la généralisation temporelle, tandis que la phase globale évalue la couverture multi-attaques.

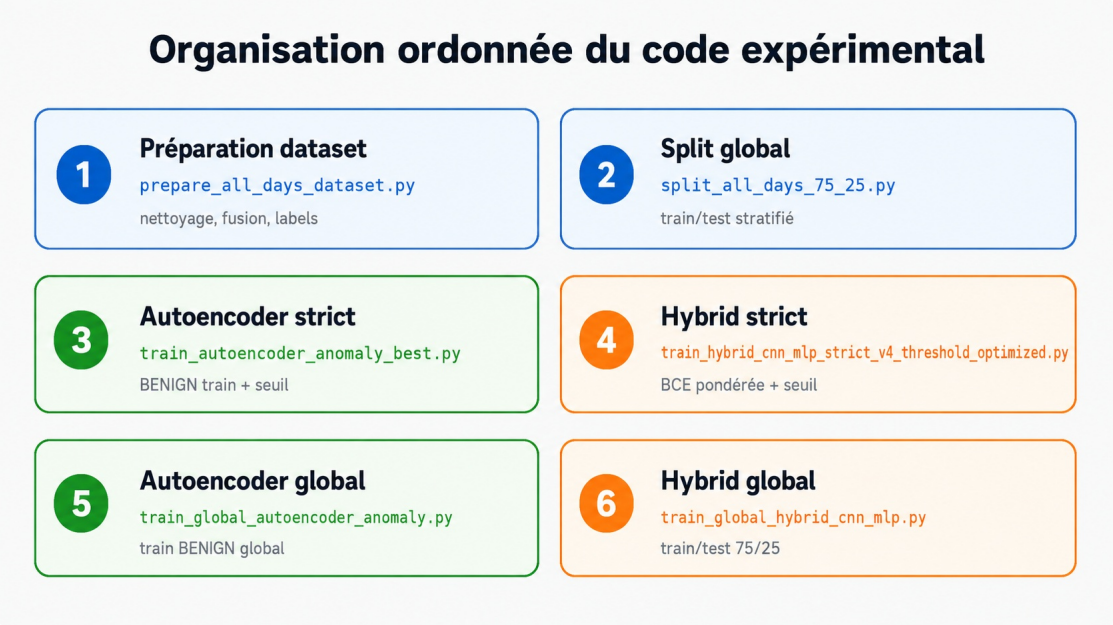

FIG. 3.2 : Organisation logique des scripts sans insertion des codes complets dans le manuscrit.

La figure 3.2 illustre la manière dont les scripts sont organisés. Les scripts de préparation produisent des fichiers propres et reproductibles. Les scripts d'entraînement utilisent ensuite ces fichiers pour ajuster les modèles, calibrer les seuils et sauvegarder les objets nécessaires à l'inférence.

3.4 Création de la Phase 1 : protocole strict

La phase stricte a été conçue pour répondre à une question précise : un modèle entraîné sur certains jours du dataset peut-il rester efficace lorsqu'il est testé sur un jour différent, contenant une attaque spécifique ? Cette phase ne cherche donc pas seulement une bonne performance globale ; elle cherche à tester la capacité du modèle à généraliser dans le temps.

3.4.1 Principe méthodologique de la Phase 1

Dans ce protocole, le fichier Friday DDoS est réservé au test final. Pour le modèle hybride supervisé, Monday, Tuesday et Wednesday sont utilisés pour l'apprentissage et la validation interne. Pour le Denoising Autoencoder, seuls les flux BENIGN de Monday et Tuesday servent à l'apprentissage du comportement normal, tandis que Wednesday sert à la calibration du seuil d'anomalie. Cette séparation évite que le modèle observe, même indirectement, les exemples utilisés pour le test final.

Algorithmiquement, la phase stricte peut être résumée ainsi :

1. préparer les fichiers nettoyés par jour avec `prepare_all_days_dataset.py` ;
2. charger les journées nécessaires au protocole strict ;
3. transformer les labels en binaire BENIGN/ATTACK tout en conservant `AttackType` pour l'analyse ;
4. ajuster la sélection de variables et la standardisation uniquement sur l'entraînement ;
5. entraîner le modèle selon sa logique propre : supervisée pour le CNN-MLP, reconstruction du BENIGN pour l'autoencodeur ;
6. choisir le seuil sur validation ou calibration ;
7. tester définitivement sur Friday DDoS.

Ce protocole correspond à une évaluation plus prudente, car le test final provient d'un jour différent. Si le modèle réussit dans ce cadre, il montre une meilleure robustesse face au décalage temporel des distributions.

3.4.2 Lien entre la séparation temporelle et le code

Le code ne doit pas mélanger les jours destinés à l'entraînement et ceux destinés au test. Le principe algorithmique est donc de construire deux ensembles distincts :

$$\mathcal{D}_{train} = \mathcal{D}_{Monday} \cup \mathcal{D}_{Tuesday} \cup \mathcal{D}_{Wednesday}, \tag{3.3}$$

$$\mathcal{D}_{test} = \mathcal{D}_{FridayDDoS}. \tag{3.4}$$

Le fragment suivant traduit directement cette idée pour le modèle hybride supervisé :

```
train_days = ["Monday_clean.csv", "Tuesday_clean.csv", "Wednesday_clean.
    csv"]
test_day   = "Friday_DDoS_clean.csv"

train_df = pd.concat([pd.read_csv(f) for f in train_days], ignore_index=
    True)
test_df  = pd.read_csv(test_day)
```

Listing 3.2: Construction conceptuelle du protocole strict supervisé.

Pour l'autoencodeur, la logique est légèrement différente : le train contient uniquement le trafic BENIGN de Monday et Tuesday, alors que Wednesday sert à calibrer le seuil. Dans les deux cas, Friday DDoS reste isolé jusqu'à la fin. Ce choix limite la fuite d'information et rend l'évaluation plus réaliste.

3.5 Création de la Phase 2 : protocole global

La phase globale répond à une autre question : quelle est la performance des modèles lorsque l'on considère l'ensemble du dataset CICIDS2017 et plusieurs familles d'attaques ? Contrairement au protocole strict, cette phase vise surtout la couverture globale et la comparaison générale des modèles.

3.5.1 Principe méthodologique de la Phase 2

Les huit fichiers CSV sont d'abord nettoyés puis fusionnés. Ensuite, un split stratifié 75 %/25 % est appliqué. La stratification est réalisée sur le type d'attaque, et non seulement sur le label binaire, afin de conserver dans le train et le test une représentation comparable de chaque famille.

Mathématiquement, si $\mathcal{C}$ représente l'ensemble des classes d'attaque, la stratification cherche à préserver :

$$P_{train}(c) \approx P_{test}(c), \quad \forall c \in \mathcal{C}. \tag{3.5}$$

Cela signifie que les distributions des classes dans le train et dans le test doivent rester proches. Sans cette contrainte, certaines attaques rares pourraient disparaître du test ou être presque absentes du train.

3.5.2 Lien entre la stratification et le code

Le fragment suivant traduit ce principe en utilisant `AttackType` comme variable de stratification :

```
train_df, test_df = train_test_split(
    df,
    test_size=0.25,
    random_state=42,
    shuffle=True,
    stratify=df["AttackType"]
)
```

Listing 3.3: Split stratifié selon le type d'attaque.

L'instruction `stratify=df["AttackType"]` représente le lien direct avec l'équation précédente. Le code impose que chaque famille d'attaque conserve approximativement la même proportion dans les deux partitions. Le paramètre `random_state=42` fixe la graine du tirage aléatoire, ce qui rend le split reproductible.

3.6 Nettoyage des données : justification mathématique et traduction code

Les données CICIDS2017 proviennent de flux réseau extraits automatiquement. Elles peuvent contenir des espaces invisibles dans les noms de colonnes, des labels mal formatés, des valeurs infinies ou des valeurs manquantes. Ces anomalies ne sont pas des attaques ; ce sont des problèmes de représentation numérique. Elles doivent donc être corrigées avant toute modélisation.

3.6.1 Problème mathématique des valeurs invalides

Un réseau de neurones apprend par descente de gradient. Pour un paramètre w, la mise à jour générale est :

$$w_{t+1} = w_t - \eta \frac{\partial \mathcal{L}}{\partial w_t}. \tag{3.6}$$

Si une entrée contient $+\infty$, $-\infty$ ou `NaN`, alors la perte $\mathcal{L}$ peut devenir indéfinie. Les gradients deviennent eux aussi indéfinis, et la mise à jour des poids n'a plus de sens numérique. C'est pourquoi toutes les valeurs non finies doivent être supprimées ou remplacées avant l'entraînement.

3.6.2 Lien avec le fragment de nettoyage

```
df.columns = df.columns.str.strip()
df["Label"] = df["Label"].astype(str).str.strip()
df.replace([np.inf, -np.inf], np.nan, inplace=True)
df.dropna(inplace=True)
```

Listing 3.4: Nettoyage minimal avant modélisation.

Chaque ligne correspond à une nécessité algorithmique précise. La première ligne garantit que les noms de colonnes sont cohérents. La deuxième harmonise les labels textuels. La troisième transforme les valeurs infinies en valeurs manquantes, et la dernière supprime les lignes qui ne peuvent pas être utilisées dans un calcul tensoriel stable.

3.7 Encodage des labels et séparation X/y

Après nettoyage, le problème est formulé comme une classification binaire. Cette étape est indispensable pour le modèle CNN-MLP, qui produit une sortie scalaire interprétée comme un score d'attaque.

3.7.1 Formulation mathématique du label binaire

Pour chaque flux x_i, le label binaire est défini par :

$$y_i = \mathbb{1}\left[\text{Label}_i \neq \text{BENIGN}\right]. \tag{3.7}$$

Ainsi, toutes les attaques sont regroupées dans la classe 1. Cette formulation correspond au comportement attendu d'un IDS : décider si un flux est normal ou suspect.

3.7.2 Lien avec le code d'encodage

```
df["AttackType"]  = df["Label"]
df["LabelBinary"] = df["Label"].apply(lambda x: 0 if x == "BENIGN"
    else 1)
```

Listing 3.5: Encodage binaire des flux réseau.

La colonne `AttackType` conserve la classe originale afin de réaliser la stratification et l'analyse multi-attaques. La colonne `LabelBinary` sert à l'entraînement binaire. Les deux colonnes n'ont donc pas le même rôle : l'une organise l'expérience, l'autre définit la cible de prédiction.

TAB. 3.4 : Colonnes exclues des variables d'entrée afin d'éviter la fuite d'information.

Colonne	Raison de l'exclusion
`Label`	Contient directement l'étiquette textuelle réelle du flux.
`LabelBinary`	Représente la classe binaire cible BENIGN/ATTACK.
`AttackType`	Sert à la stratification et à l'analyse par famille, mais ne doit pas guider la prédiction.
`SourceFile`	Indique l'origine temporelle du flux et peut créer un raccourci artificiel lié au jour de capture.

3.8 Prétraitement numérique : sélection et standardisation

Avant d'entraîner les modèles, la matrice de caractéristiques doit être préparée. Cette étape est commune au CNN-MLP et à l'Autoencoder, même si l'Autoencoder reçoit en plus une transformation logarithmique signée.

3.8.1 Sélection des caractéristiques numériques

Les modèles neuronaux ne manipulent pas directement des chaînes de caractères. La matrice d'entrée doit donc contenir uniquement des colonnes numériques :

$$X \in \mathbb{R}^{n \times d}. \tag{3.8}$$

Le fragment suivant traduit cette contrainte :

```
y = df["LabelBinary"].astype(int)
X = df.drop(columns=["Label", "AttackType", "LabelBinary", "
    SourceFile"], errors="ignore")
X = X.select_dtypes(include=[np.number])
```

Listing 3.6: Extraction de la matrice numérique.

Les colonnes textuelles sont retirées, car elles servent à documenter les données mais ne sont pas directement exploitables par le modèle. La variable X contient donc les caractéristiques, et y contient les labels.

3.8.2 Suppression des colonnes constantes

Une caractéristique de variance nulle vérifie :

$$\mathrm{Var}(X_j) = 0. \tag{3.9}$$

Cela signifie que cette colonne prend la même valeur pour tous les flux. Elle ne peut donc pas aider à distinguer un trafic normal d'une attaque. Elle est supprimée par :

```
selector = VarianceThreshold(threshold=0.0)
X_train = selector.fit_transform(X_train)
X_val   = selector.transform(X_val)
X_test  = selector.transform(X_test)
```

Listing 3.7: Suppression des caractéristiques constantes.

Le `fit` est appliqué uniquement sur l'entraînement, car le test doit rester inconnu pendant la construction du pipeline. Le même sélecteur est ensuite appliqué à la validation et au test.

3.8.3 Standardisation et prévention du data leakage

La standardisation transforme chaque variable X_j selon :

$$X'_j = \frac{X_j - \mu_j}{\sigma_j}, \tag{3.10}$$

où μ_j et σ_j sont estimés uniquement sur l'ensemble d'entraînement. Cette règle évite que le test influence les statistiques utilisées par le modèle.

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val   = scaler.transform(X_val)
X_test  = scaler.transform(X_test)
```

Listing 3.8: Standardisation ajustée uniquement sur le train.

Si `fit_transform` était appliqué sur le test, le modèle bénéficierait d'une information indirecte sur la distribution future. Ce phénomène est appelé *data leakage*. Il rendrait les résultats artificiellement meilleurs et moins fiables.

3.9 Denoising Autoencoder : lien entre mathématiques, algorithme et implémentation

L'Autoencoder est utilisé comme détecteur d'anomalies. Contrairement au CNN-MLP, il n'apprend pas directement à distinguer les classes. Il apprend à reconstruire le trafic normal. Un flux est considéré comme suspect si sa reconstruction est mauvaise.

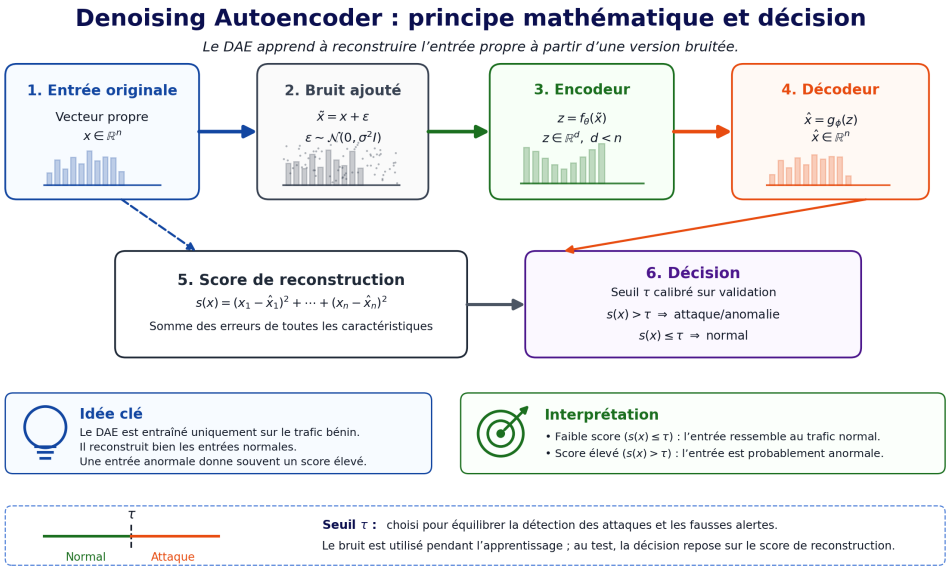

FIG. 3.3 : Principe mathématique et algorithmique du Denoising Autoencoder.

3.9.1 Pourquoi entraîner l'Autoencoder uniquement sur les flux BENIGN ?

Soit f_θ l'Autoencoder. Son objectif est d'apprendre :

$$f_\theta(x) \approx x \tag{3.11}$$

pour les flux normaux. Si le modèle voit aussi des attaques pendant l'entraînement, il peut apprendre à les reconstruire correctement. Dans ce cas, l'erreur de reconstruction ne permet plus de distinguer les anomalies. C'est pourquoi l'ensemble d'entraînement de l'Autoencoder est limité à :

$$\mathcal{D}_{AE}^{train} = \{x_i \mid y_i = 0\}. \tag{3.12}$$

Le code correspondant applique simplement un filtre sur les labels :

```
1    X_train_ae = X_train[y_train == 0]
```

Listing 3.9: Sélection du trafic normal pour l'Autoencoder.

Cette ligne est directement liée à la définition de $\mathcal{D}_{AE}^{train}$. Elle force le modèle à apprendre uniquement la structure du trafic normal.

3.9.2 Transformation logarithmique signée

Certaines variables réseau possèdent de très grandes amplitudes. Pour limiter leur domination dans l'erreur quadratique, on applique :

$$g(x) = \text{sign}(x) \log(1 + |x|). \tag{3.13}$$

Cette transformation compresse les grandes valeurs tout en conservant le signe. Elle est particulièrement utile pour l'Autoencoder, car sa décision dépend directement de l'erreur MSE sur les caractéristiques reconstruites.

```
def signed_log1p(data):
    return np.sign(data) * np.log1p(np.abs(data))
```

Listing 3.10: Transformation logarithmique signée.

Le fragment implémente exactement la fonction $g(x)$. Il ne s'agit donc pas d'un choix purement technique, mais d'une transformation mathématique visant à stabiliser la mesure de reconstruction.

3.9.3 Architecture et compression latente

L'Autoencoder est composé d'un encodeur et d'un décodeur :

$$z = E_\theta(x), \quad \hat{x} = D_\theta(z). \tag{3.14}$$

Le vecteur z représente l'espace latent. Dans ce travail, cet espace est volontairement réduit afin de forcer le modèle à apprendre une représentation compacte du trafic normal.

```
encoder = nn.Sequential(
    nn.Linear(input_dim, 128), nn.ReLU(),
    nn.Linear(128, 64), nn.ReLU(),
    nn.Linear(64, 16), nn.ReLU(),
    nn.Linear(16, 4)
)
```

Listing 3.11: Goulot d'étranglement de l'Autoencoder.

Le passage de `input_dim` à 4 dimensions représente le goulot d'étranglement. Un flux normal peut être reconstruit à partir de cette représentation compacte. En revanche, un flux d'attaque contient souvent des relations différentes entre variables ; il est donc moins bien reconstruit.

3.9.4 Principe du débruitage

Pendant l'entraînement, le modèle reçoit une version bruitée de l'entrée :

$$\tilde{x} = x + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2), \tag{3.15}$$

mais il doit reconstruire l'entrée propre x. Ce principe évite que le réseau apprenne une simple fonction identité.

```
x_noisy = x_clean + noise_std * torch.randn_like(x_clean)
loss = criterion(model(x_noisy), x_clean)
```

Listing 3.12: Ajout du bruit pendant l'apprentissage.

La première ligne applique la perturbation $\tilde{x}$, tandis que la deuxième impose au modèle de reconstruire x. Ce lien entre bruit et reconstruction est le cœur du Denoising Autoencoder.

3.9.5 Erreur de reconstruction et décision

Après entraînement, chaque flux reçoit un score d'anomalie :

$$e(x_i) = \frac{1}{d} \sum_{j=1}^{d} (x_{ij} - \hat{x}_{ij})^2 . \tag{3.16}$$

La règle de décision est :

$$\hat{y}_i = \begin{cases} 1, & \text{si } e(x_i) \geq \tau, \\ 0, & \text{sinon.} \end{cases} \tag{3.17}$$

```
x_hat = model(x)
errors = torch.mean((x - x_hat) ** 2, dim=1)
pred = (errors >= threshold).cpu().numpy().astype(int)
```

Listing 3.13: Score d'anomalie par erreur de reconstruction.

Le code suit directement les équations : `errors` correspond à $e(x_i)$ et `threshold` correspond à τ.

3.10 Hybrid CNN-MLP : lien entre mathématiques, algorithme et implémentation

Le modèle hybride CNN-MLP est un classifieur supervisé. Il reçoit un vecteur de caractéristiques et produit un logit, ensuite transformé en probabilité d'attaque par une sigmoïde.

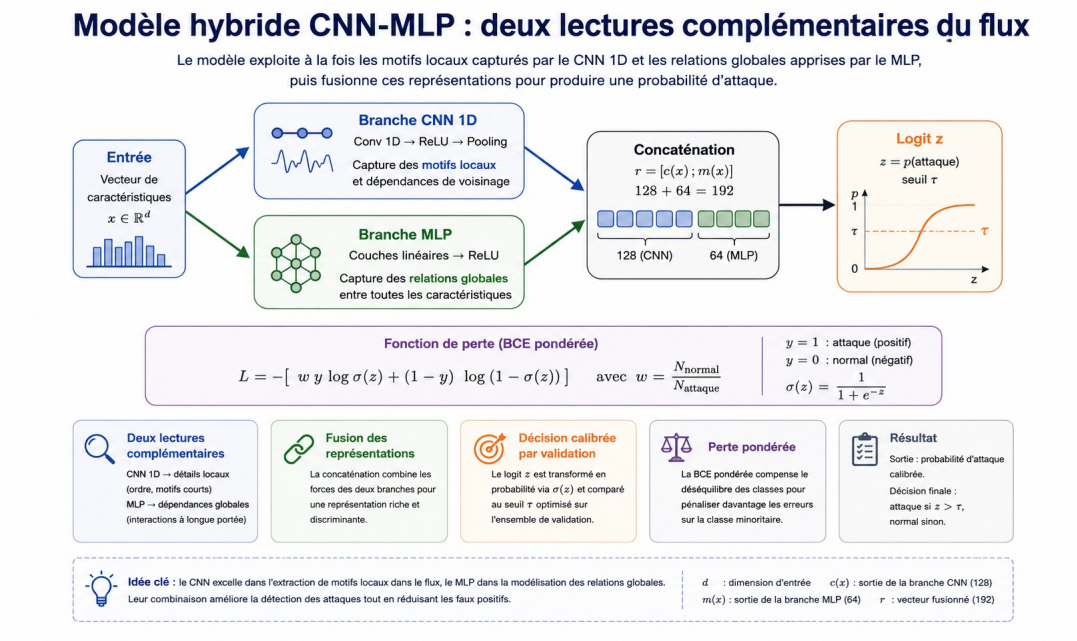

FIG. 3.4 : Principe mathématique et algorithmique du modèle hybride CNN-MLP.

3.10.1 Pourquoi combiner CNN et MLP ?

La branche CNN cherche des motifs locaux dans le vecteur de caractéristiques. Même si les données ne sont pas des images, certaines variables proches dans l'ordre du dataset peuvent porter des relations utiles. Une convolution 1D calcule :

$$h_k(t) = \phi\left(b_k + \sum_{r=0}^{K-1} w_{kr} x_{t+r}\right), \tag{3.18}$$

où K est la taille du noyau, w_{kr} les poids du filtre et ϕ la fonction d'activation.

La branche MLP, de son côté, traite toutes les caractéristiques simultanément :

$$m = \phi(Wx + b). \tag{3.19}$$

Elle apprend donc des relations globales entre les variables.

Les deux représentations sont ensuite concaténées :

$$r(x) = [c(x); m(x)], \tag{3.20}$$

où $c(x)$ est la sortie CNN et $m(x)$ la sortie MLP.

3.10.2 Lien avec l'implémentation des deux branches

```
cnn_features = self.cnn_branch(x)              # motifs locaux
mlp_features = self.mlp_branch(x.squeeze(1))   # relations globales
features = torch.cat([cnn_features, mlp_features], dim=1)
logit = self.classifier(features)
```

Listing 3.14: Structure conceptuelle des deux branches.

Le code suit exactement la formulation $r(x) = [c(x); m(x)]$. La concaténation ne sert pas seulement à augmenter la dimension ; elle réunit deux lectures différentes du même flux.

3.10.3 Forme de l'entrée pour la convolution 1D

PyTorch attend une entrée de forme :

$$(N, C, L), \tag{3.21}$$

où N est le nombre d'exemples, C le nombre de canaux et L la longueur de la séquence. Ici, le vecteur de caractéristiques est considéré comme une séquence à un seul canal :

$$X \in \mathbb{R}^{N \times 1 \times d}. \tag{3.22}$$

```
X_train_tensor = torch.tensor(X_train, dtype=torch.float32).unsqueeze(1)
```

Listing 3.15: Mise en forme de l'entrée pour Conv1D.

L'instruction `unsqueeze(1)` ajoute la dimension canal. Sans cette dimension, la couche `Conv1d` ne peut pas interpréter correctement le vecteur d'entrée.

3.10.4 Perte supervisée pondérée

Le dataset est déséquilibré : les flux normaux sont majoritaires. Pour éviter que le modèle apprenne à prédire uniquement la classe BENIGN, on pondère la perte de la classe attaque. La perte BCE pondérée est :

$$\mathcal{L} = -\frac{1}{n} \sum_{i=1}^{n} \left[w y_i \log(\sigma(z_i)) + (1 - y_i) \log(1 - \sigma(z_i)) \right], \tag{3.23}$$

où z_i est le logit, σ la sigmoïde et w le poids de la classe positive.

```
n_benign = (y_train == 0).sum()
n_attack = (y_train == 1).sum()
pos_weight = torch.tensor([n_benign / n_attack], dtype=torch.float32)
criterion = nn.BCEWithLogitsLoss(pos_weight=pos_weight.to(device))
```

Listing 3.16: Pondération de la classe attaque.

Le rapport entre le nombre de flux normaux et le nombre de flux attaques détermine l'importance accordée aux erreurs sur la classe attaque. Le rapport exact dépend de la partition d'apprentissage après nettoyage et stratification : le modèle pénalise donc plus fortement les erreurs commises sur la classe malveillante lorsque celle-ci est minoritaire. Ce choix n'est pas décoratif ; sans cette pondération, les premiers essais favorisaient trop facilement la classe BENIGN majoritaire.

3.10.5 Sigmoïde et probabilité d'attaque

Le classifieur produit un logit z. La probabilité d'attaque est obtenue par :

$$p = \sigma(z) = \frac{1}{1 + e^{-z}}. \tag{3.24}$$

La décision dépend ensuite d'un seuil τ :

$$\hat{y} = \mathbb{1}[p \geq \tau]. \tag{3.25}$$

```
prob = torch.sigmoid(logits).cpu().numpy()
pred = (prob >= threshold).astype(int)
```

Listing 3.17: Transformation du logit en décision.

Le fragment montre que le modèle ne décide pas directement avec le logit. Il transforme d'abord la sortie en probabilité, puis applique le seuil retenu sur validation.

3.11 Calibration du seuil : lien commun entre les deux modèles

Les deux modèles produisent des scores de nature différente. L'Autoencoder produit une erreur de reconstruction ; le CNN-MLP produit une probabilité d'attaque. Dans les deux cas, une décision binaire exige un seuil.

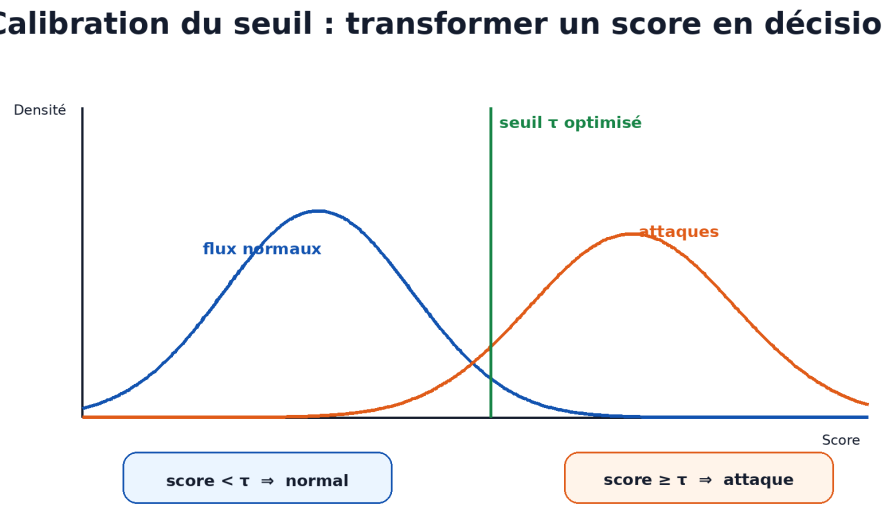

FIG. 3.5 : Calibration du seuil à partir d'un score continu.

3.11.1 Pourquoi ne pas fixer un seuil arbitraire ?

Un seuil arbitraire peut favoriser excessivement la précision ou le rappel. Dans un IDS, le rappel de la classe attaque est particulièrement important, car il indique la proportion d'attaques réellement détectées. Pour équilibrer précision et rappel, le seuil est choisi en maximisant le F1-score :

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \tag{3.26}$$

3.11.2 Calibration du seuil pour l'Autoencoder

Pour l'Autoencoder, les seuils candidats sont extraits des percentiles des erreurs de reconstruction :

$$\tau_q = \text{Percentile}_q(e(x)). \tag{3.27}$$

```
best_threshold, best_score = None, -1
for q in np.arange(70, 99.5, 0.5):
    threshold = np.percentile(errors_val, q)
    pred = (errors_val >= threshold).astype(int)
    score = f1_score(y_val, pred, pos_label=1)
    if score > best_score:
        best_threshold, best_score = threshold, score
```

Listing 3.18: Recherche du seuil AE par percentiles.

Chaque percentile propose un compromis différent. Un seuil bas détecte davantage d'attaques, mais produit plus de faux positifs ; un seuil haut réduit les fausses alertes, mais

risque de manquer des attaques. La calibration est donc réalisée sur validation ou calibration, jamais sur le test final.

3.11.3 Calibration du seuil pour le CNN-MLP

Pour le CNN-MLP, le score est une probabilité. Le seuil est donc recherché dans l'intervalle $[0, 1]$:

```
best_threshold, best_score = None, -1
for t in np.arange(0.10, 0.91, 0.01):
    pred = (y_prob_val >= t).astype(int)
    score = f1_score(y_val, pred, pos_label=1)
    if score > best_score:
        best_threshold, best_score = t, score
```

Listing 3.19: Recherche du seuil CNN-MLP par grille.

Ce fragment est lié à la règle $\hat{y} = \mathbb{1}[p \geq \tau]$. Le seuil final n'est pas choisi manuellement ; il est déterminé par la validation.

3.12 Évaluation finale des modèles

Après calibration du seuil, les modèles sont évalués sur un test final qui n'a pas été utilisé pendant l'entraînement ni pendant le réglage du seuil. L'évaluation repose sur la matrice de confusion :

Tab. 3.5 : Structure de la matrice de confusion binaire.

	Prédit BENIGN	**Prédit ATTACK**
Réel BENIGN	TN	FP
Réel ATTACK	FN	TP

Les métriques utilisées sont :

$$Precision = \frac{VP}{VP + FP}, \tag{3.28}$$

$$Recall = \frac{VP}{VP + FN}, \tag{3.29}$$

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \tag{3.30}$$

Dans le contexte IDS, l'accuracy seule n'est pas suffisante. Un modèle peut obtenir une accuracy élevée s'il prédit majoritairement la classe normale, surtout lorsque les classes sont déséquilibrées. Le rappel attaque et le F1-score sont donc plus informatifs pour mesurer la capacité réelle de détection.

```
report = classification_report(y_test, test_pred, target_names=["BENIGN",
    "ATTACK"])
bal_acc = balanced_accuracy_score(y_test, test_pred)
```

Listing 3.20: Évaluation commune après décision par seuil.

Le rapport de classification donne la précision, le rappel et le F1-score pour chaque classe. La balanced accuracy complète cette lecture en tenant compte du déséquilibre des classes.

3.13 Sauvegarde des modèles et reproductibilité

Un modèle entraîné ne suffit pas pour refaire une prédiction correcte. Il faut aussi sauvegarder les objets de prétraitement ajustés sur le train : le sélecteur de variables, le scaler et le seuil de décision. Sinon, les nouvelles données ne seront pas transformées de la même manière que les données d'entraînement.

```
torch.save(model.state_dict(), model_path)
joblib.dump(selector, selector_path)
joblib.dump(scaler, scaler_path)
np.save(threshold_path, np.array([threshold]))
```

Listing 3.21: Sauvegarde des objets nécessaires à l'inférence.

Cette sauvegarde garantit que le pipeline complet peut être réutilisé. Le modèle seul contient les poids appris, mais le scaler contient les moyennes et écarts-types utilisés pendant l'entraînement. Ces paramètres doivent rester identiques en phase de test ou de déploiement.

3.14 Conclusion

Ce chapitre a présenté l'implémentation de manière liée : chaque fragment de code a été introduit après son principe méthodologique et sa justification mathématique. La création de la phase stricte a été reliée à l'évaluation temporelle, tandis que la phase globale a été reliée à la couverture multi-attaques. Le nettoyage a été justifié par la nécessité de garantir des gradients définis. La standardisation a été expliquée comme une condition de stabilité numérique et de prévention du data leakage.

L'Autoencoder a été présenté comme un modèle de reconstruction du trafic normal, où l'anomalie est détectée par l'erreur MSE. Le modèle Hybrid CNN-MLP a été présenté comme un classifieur supervisé combinant une lecture locale du vecteur par convolution 1D et une lecture globale par MLP. Enfin, la calibration du seuil a été expliquée comme une étape centrale de passage d'un score continu vers une décision IDS binaire.

Ainsi, ce chapitre établit clairement la correspondance entre les choix mathématiques, les algorithmes et leur traduction en code, sans insérer les scripts complets dans le corps du mémoire. Les résultats obtenus avec ces protocoles sont analysés dans le chapitre suivant.

Chapitre 4 : Expérimentations, résultats et discussion

4.1 Introduction

Ce chapitre présente et interprète les résultats expérimentaux obtenus avec les deux modèles étudiés dans cette étude : le *Denoising Autoencoder* et le *Hybrid CNN-MLP*. L'objectif n'est pas de juxtaposer des scores, mais d'expliquer leur signification opérationnelle pour un système de détection d'intrusions réseau. Les résultats sont donc analysés à travers la précision, le rappel, le F1-score, la balanced accuracy, les matrices de confusion, les faux positifs et les faux négatifs.

Deux protocoles complémentaires sont utilisés. Le premier est un **protocole strict par séparation temporelle** : le test final est réalisé sur Friday DDoS, séparé des journées utilisées pour l'entraînement ou la calibration. Le second est un **protocole global** : les fichiers CSV principaux de *CICIDS2017* sont fusionnés, nettoyés puis divisés selon un split stratifié 75 %/25 %. La figure 4.1 résume la logique de lecture de ces deux phases.

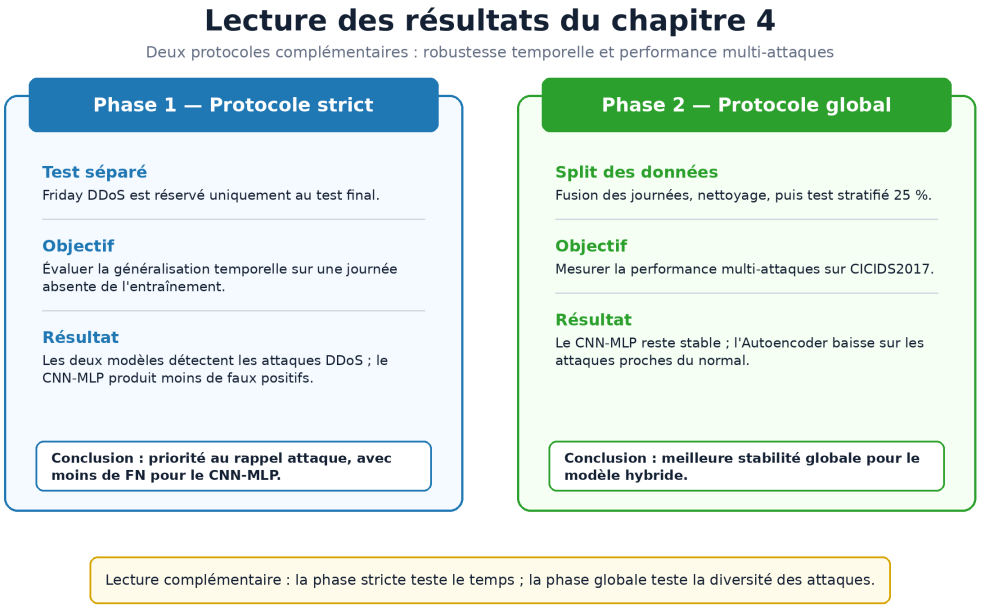

Fig. 4.1 : Synthèse des deux protocoles expérimentaux et de leur rôle dans l'évaluation.

Le protocole strict mesure surtout la robustesse face à un décalage temporel entre les

données d'apprentissage et les données de test. Le protocole global mesure la performance moyenne sur une plus grande diversité de scénarios d'attaque. Cette distinction est essentielle : un score élevé dans un split global ne garantit pas nécessairement une bonne généralisation temporelle, et un bon résultat sur Friday DDoS ne suffit pas à prouver la robustesse sur toutes les familles d'attaques.

4.2 Rappel des métriques d'évaluation

Dans une matrice de confusion binaire, les flux BENIGN correctement classés sont des vrais négatifs (VN), les attaques correctement détectées sont des vrais positifs (VP), les flux BENIGN signalés à tort comme attaques sont des faux positifs (FP), et les attaques classées à tort comme BENIGN sont des faux négatifs (FN). Dans un IDS, les FN sont généralement les erreurs les plus critiques, car ils correspondent à des intrusions non détectées. Les FP restent néanmoins importants, car ils augmentent la charge d'analyse humaine.

Tab. 4.1 : Lecture opérationnelle des erreurs dans un IDS.

Terme	**Cas**	**Interprétation opérationnelle**
Vrai positif (VP)	ATTACK → ATTACK	Une attaque est détectée correctement.
Vrai négatif (VN)	BENIGN → BENIGN	Un flux normal n'est pas signalé inutilement.
Faux positif (FP)	BENIGN → ATTACK	Une fausse alerte est produite ; elle augmente le bruit opérationnel.
Faux négatif (FN)	ATTACK → BENIGN	Une attaque n'est pas détectée ; c'est l'erreur la plus critique.

Les métriques principales sont définies par :

$$Accuracy = \frac{VP + VN}{VP + VN + FP + FN}, \tag{4.1}$$

$$Precision_{attaque} = \frac{VP}{VP + FP}, \qquad Rappel_{attaque} = \frac{VP}{VP + FN}, \tag{4.2}$$

$$F1_{attaque} = 2 \times \frac{Precision_{attaque} \times Rappel_{attaque}}{Precision_{attaque} + Rappel_{attaque}}. \tag{4.3}$$

La balanced accuracy est également utile lorsque les classes sont déséquilibrées. Elle correspond à la moyenne des rappels par classe :

$$Balanced\ Accuracy = \frac{1}{2}\left(Rappel_{BENIGN} + Rappel_{ATTACK}\right). \tag{4.4}$$

Dans ce travail, le rappel attaque est prioritaire, mais il doit toujours être interprété avec

les FP. Un rappel attaque maximal peut être obtenu au prix d'un nombre excessif de fausses alertes ; inversement, une précision élevée peut masquer des attaques manquées.

4.3 Phase 1 : protocole strict par séparation temporelle

4.3.1 Organisation du protocole

La phase stricte isole Friday DDoS pour le test final. Pour le *Hybrid CNN-MLP*, les journées Monday, Tuesday et Wednesday servent à l'apprentissage et à la validation interne. Pour le *Denoising Autoencoder*, l'apprentissage est réalisé sur les flux BENIGN de Monday et Tuesday, tandis que Wednesday est utilisé pour calibrer le seuil d'anomalie. Cette différence reflète la nature des deux modèles : le modèle hybride est supervisé, alors que l'autoencodeur apprend d'abord le comportement normal.

Tab. 4.2 : Organisation de la phase stricte.

Élément	Hybrid CNN-MLP	Denoising Autoencoder
Apprentissage	Monday, Tuesday, Wednesday avec labels binaires	Flux BENIGN de Monday et Tuesday
Validation / calibration	Split interne sur l'ensemble d'apprentissage	Wednesday, utilisé pour choisir le seuil
Test final	Friday DDoS uniquement	Friday DDoS uniquement
Objet sauvegardé	Modèle, sélecteur, scaler, seuil sigmoïde	Modèle, sélecteur, scaler, seuil d'erreur
But	Apprendre une frontière BENIGN/ATTACK	Mesurer l'écart au comportement normal

Cette séparation limite la fuite d'information : le sélecteur de variance, le scaler et les seuils sont ajustés sans utiliser les données du test final. Friday DDoS n'intervient qu'une seule fois, au moment de l'évaluation finale.

La figure 4.2 montre que le test strict contient 225 711 flux, dont 97 686 flux BENIGN et 128 025 flux ATTACK. Cette répartition permet d'évaluer les modèles dans un scénario temporel séparé, où l'objectif principal est de vérifier que les attaques DDoS du vendredi sont détectées sans avoir été utilisées pendant l'entraînement.

4.3.2 Résultats du Hybrid CNN-MLP

Le *Hybrid CNN-MLP* utilise une sortie sigmoïde et un seuil calibré sur validation. En phase stricte, le seuil retenu est $\tau = 0{,}90$. Ce seuil élevé indique que le modèle ne classe un flux comme attaque que lorsque le score supervisé est très affirmé.

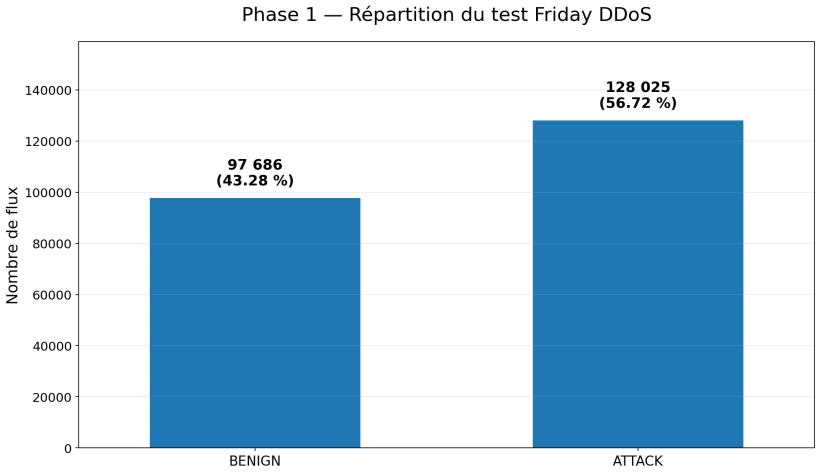

FIG. 4.2 : Répartition binaire des flux dans le test strict Friday DDoS.

TAB. 4.3 : Résultats du Hybrid CNN-MLP sur Friday DDoS.

Classe	Précision	Rappel	F1-score	Support
BENIGN	1,00	0,79	0,88	97 686
ATTACK	0,86	1,00	0,92	128 025
Accuracy		0,91		225 711
Balanced accuracy		0,895		225 711

Le rappel attaque de $1{,}00$ signifie qu'aucune attaque DDoS du test strict n'est manquée. La limite principale apparaît du côté BENIGN : une partie des flux normaux est signalée comme attaque, ce qui génère des fausses alertes. Pour un IDS, ce comportement peut être toléré lorsque la priorité est d'éviter les attaques non détectées, mais il doit être surveillé en contexte réel.

4.3.3 Résultats du Denoising Autoencoder

Le *Denoising Autoencoder* décide à partir de l'erreur de reconstruction. Dans cette phase, le seuil retenu est $\tau_{AE} = 0{,}0969$. Le modèle ne reçoit pas d'exemples d'attaques pendant l'apprentissage principal ; il apprend à reconstruire le trafic BENIGN, puis signale les flux qui s'en écartent.

TAB. 4.4 : Résultats du Denoising Autoencoder sur Friday DDoS.

Classe	Précision	Rappel	F1-score	Support
BENIGN	0,99	0,65	0,78	97 686
ATTACK	0,79	1,00	0,88	128 025
Accuracy		0,85		225 711
Balanced accuracy		0,825		225 711

L'autoencodeur détecte également les attaques DDoS avec un rappel maximal, mais au prix d'un rappel BENIGN plus faible. Cela confirme que le seuil d'anomalie est plus sensible : il protège contre les FN, mais augmente les FP.

4.3.4 Comparaison détaillée en phase stricte

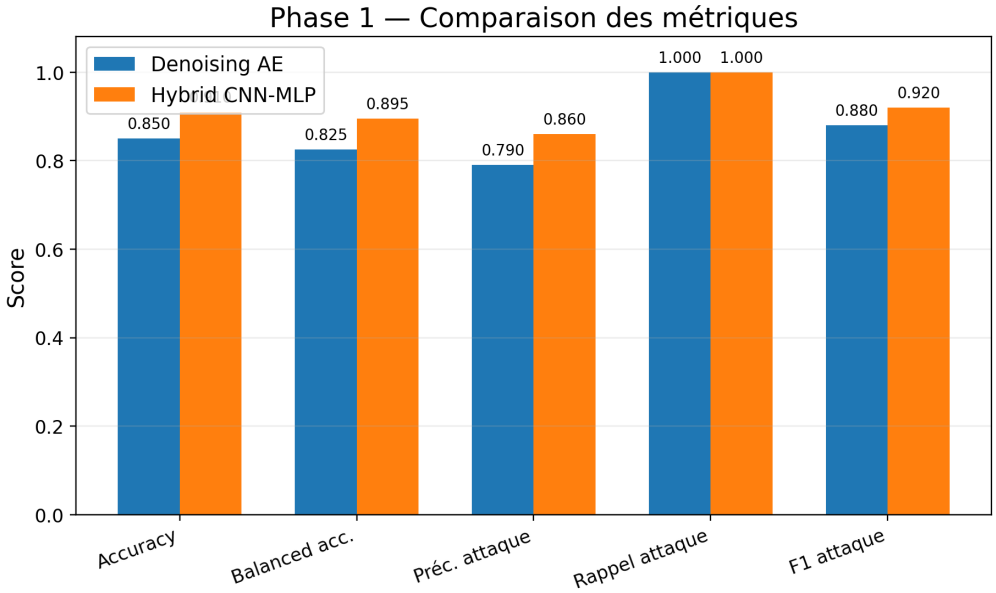

Fig. 4.3 : Comparaison des métriques principales en phase stricte sur Friday DDoS.

La figure 4.3 met en évidence que les deux modèles atteignent un rappel attaque maximal en phase stricte. La différence ne se situe donc pas dans la détection des attaques DDoS, mais dans la qualité de séparation du trafic BENIGN. Le Hybrid CNN-MLP obtient une meilleure accuracy, une meilleure précision attaque et un meilleur F1-score, ce qui traduit une meilleure maîtrise des fausses alertes.

Tab. 4.5 : Synthèse des faux positifs et faux négatifs en phase stricte.

Modèle	VN	FP	FN	VP	Lecture
Hybrid CNN-MLP	≈76 800	≈20 886	0	128 025	Aucun DDoS manqué, mais fausses alertes modérées.
Denoising Autoencoder	≈63 496	≈34 190	0	128 025	Aucun DDoS manqué, mais plus de trafic BENIGN signalé.

Les valeurs du tableau 4.5 sont cohérentes avec les supports et les rappels arrondis des rapports de classification. Elles montrent que la différence entre les deux modèles en phase stricte ne se situe pas dans les FN, mais dans le volume de FP. Le *Hybrid CNN-MLP* est donc plus équilibré pour Friday DDoS.

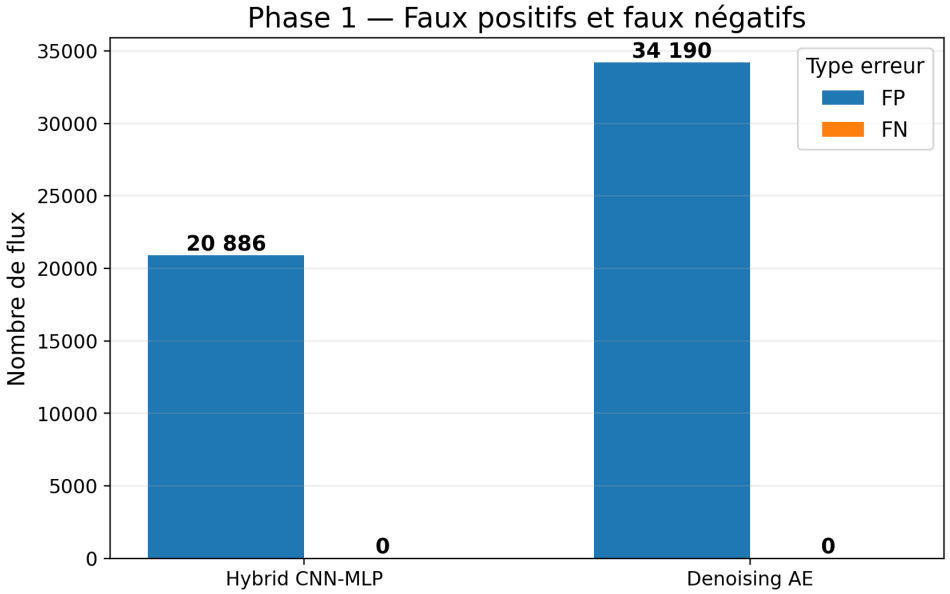

FIG. 4.4 : Comparaison des faux positifs et faux négatifs en phase stricte.

La figure 4.4 confirme que les deux modèles ne manquent aucune attaque DDoS dans le test strict. En revanche, le Denoising Autoencoder produit environ 34 190 faux positifs, contre environ 20 886 pour le Hybrid CNN-MLP. Le gain du modèle hybride est donc principalement opérationnel : il réduit le nombre d'alertes inutiles tout en conservant un rappel attaque maximal.

4.4 Phase 2 : protocole global 75 %/25 %

4.4.1 Organisation du protocole global

Dans la phase globale, les fichiers principaux de *CICIDS2017* sont nettoyés, fusionnés, puis divisés en 75 % pour l'entraînement et 25 % pour le test. La stratification est réalisée sur `AttackType`, ce qui préserve autant que possible la représentation des familles d'attaques dans les deux ensembles.

TAB. 4.6 : Répartition binaire du test global 25 %.

Classe	Nombre de flux	Pourcentage
BENIGN	567 830	80,32 %
ATTACK	139 139	19,68 %
Total	706 969	100,00 %

Ce test est déséquilibré : les flux BENIGN représentent plus de quatre cinquièmes des observations. L'accuracy seule serait donc insuffisante. Le rappel attaque, le F1-score, la ba-

lanced accuracy et la comparaison FP/FN sont indispensables pour juger la pertinence d'un modèle IDS.

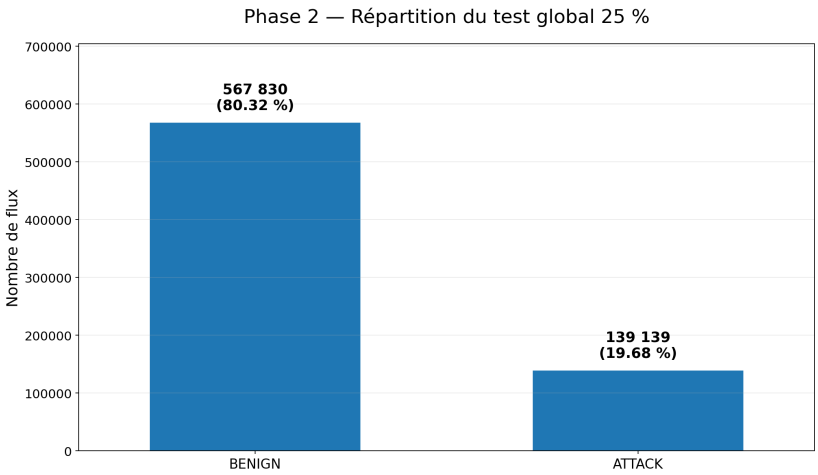

FIG. 4.5 : Répartition binaire des flux dans le test global 25 %.

La figure 4.5 montre le déséquilibre du test global : les flux BENIGN représentent 80,32 % des observations, contre 19,68 % pour les flux d'attaque. Ce déséquilibre justifie l'utilisation du rappel attaque, du F1-score et de la balanced accuracy en complément de l'accuracy.

4.4.2 Résultats du Hybrid CNN-MLP

Le seuil calibré du *Hybrid CNN-MLP* dans le protocole global est $\tau = 0{,}40$. La baisse du seuil par rapport à la phase stricte s'explique par le changement de distribution : le protocole global contient davantage de familles d'attaques et le seuil optimal de validation privilégie le rappel attaque.

TAB. 4.7 : Résultats du Hybrid CNN-MLP sur le test global.

Classe	Précision	Rappel	F1-score	Support
BENIGN	1,000	0,982	0,991	567 830
ATTACK	0,930	1,000	0,964	139 139
Accuracy		0,985		706 969
Balanced accuracy		0,991		706 969

La matrice de confusion montre $FN = 0$ et $FP = 10\,475$. Ce résultat est remarquable ; il ne doit donc pas être interprété comme une preuve absolue de perfection du modèle. Elle signifie seulement qu'aucune attaque du test global n'a été classée BENIGN dans ce découpage précis. Le résultat est crédible à condition que le test n'ait pas servi au prétraitement, que les colonnes de label aient été exclues des entrées et que le seuil ait été choisi sur validation

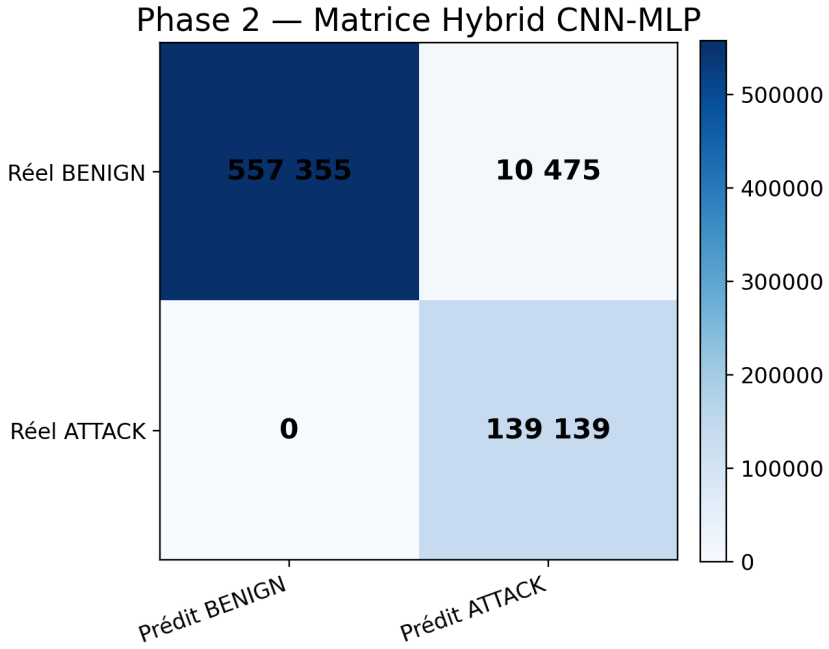

FIG. 4.6 : Matrice de confusion du Hybrid CNN-MLP sur le test global.

plutôt que sur le test. Ces conditions sont intégrées dans le protocole expérimental décrit au chapitre précédent. Il faut toutefois rester prudent : un autre dataset, une capture réelle ou une attaque plus discrète pourraient réintroduire des faux négatifs.

4.4.3 Résultats du Denoising Autoencoder

Pour l'autoencodeur global, le seuil retenu correspond au percentile 97 des erreurs de reconstruction observées sur le trafic normal d'entraînement, soit $\tau_{AE} = 0{,}1112$. Ce choix favorise une réduction des fausses alertes par rapport à des seuils plus bas, mais il peut laisser passer des attaques proches du comportement BENIGN.

TAB. 4.8 : Seuils candidats du Denoising Autoencoder global.

Percentile	Erreur de reconstruction
p_{90}	0,0540
p_{95}	0,0817
p_{97}	0,1112
p_{99}	0,2000

L'autoencodeur obtient une bonne reconnaissance du trafic BENIGN, mais manque 45 916 attaques. Cette faiblesse est cohérente avec son principe : les attaques dont les caractéristiques statistiques restent proches du trafic normal peuvent produire une erreur de reconstruction insuffisante pour dépasser le seuil. Ce comportement explique pourquoi

Tab. 4.9 : Résultats du Denoising Autoencoder sur le test global.

Classe	Précision	Rappel	F1-score	Support
BENIGN	0,923	0,970	0,946	567 830
ATTACK	0,845	0,670	0,747	139 139
Accuracy		0,911		706 969
Balanced accuracy		0,820		706 969

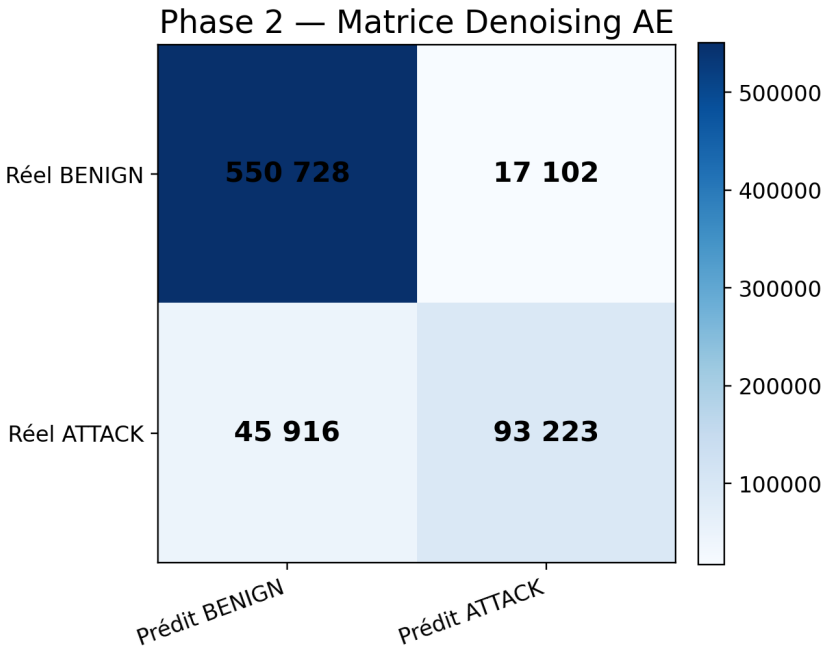

Fig. 4.7 : Matrice de confusion du Denoising Autoencoder sur le test global.

l'autoencodeur peut être utile comme outil de surveillance complémentaire, mais plus fragile comme moteur principal de classification IDS.

4.5 Comparaison globale et analyse par familles d'attaques

4.5.1 Comparaison des métriques principales

TAB. 4.10 : Comparaison synthétique des performances globales.

Modèle	Accuracy	Balanced acc.	Préc. attaque	Rappel attaque	F1 attaque
Denoising Autoencoder	0,911	0,820	0,845	0,670	0,747
Hybrid CNN-MLP	0,985	0,991	0,930	1,000	0,964

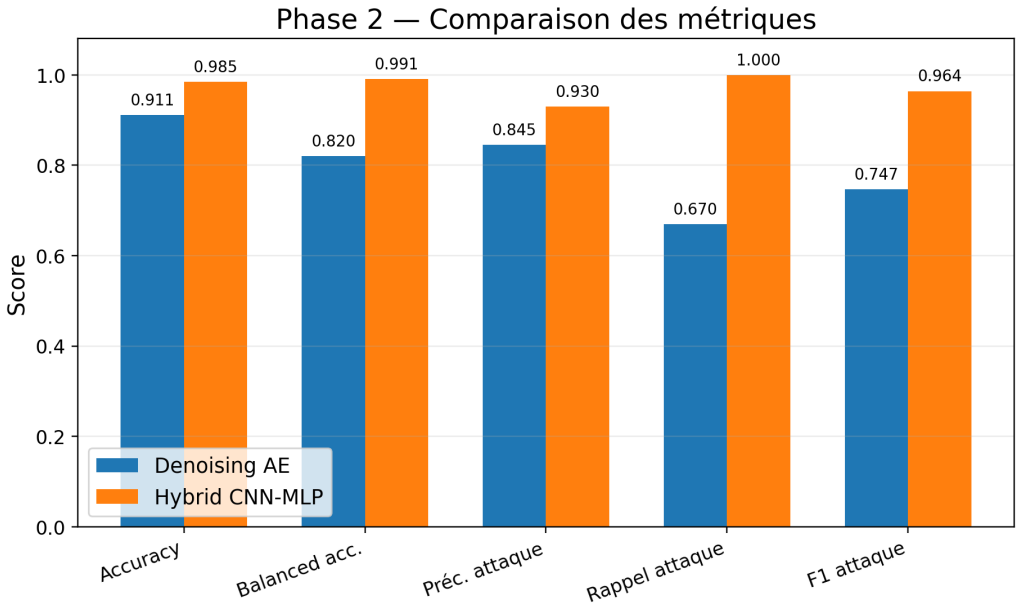

FIG. 4.8 : Comparaison globale des métriques principales sur le test 25 %.

La figure 4.8 montre que le Hybrid CNN-MLP dépasse le Denoising Autoencoder sur toutes les métriques globales. L'écart le plus important concerne le rappel attaque, qui passe de 0,670 pour l'autoencodeur à 1,000 pour le modèle hybride. Cette différence signifie que l'autoencodeur manque 45 916 attaques dans le test global, alors que le modèle hybride n'en manque aucune dans ce découpage expérimental.

Tab. 4.11 : Erreurs globales obtenues à partir des matrices de confusion.

Modèle	VN	FP	FN	VP	Erreur dominante
Hybrid CNN-MLP	557 355	10 475	0	139 139	FP limités, aucun FN dans ce split.
Denoising Autoencoder	550 728	17 102	45 916	93 223	FN nombreux sur les attaques proches du comportement normal.

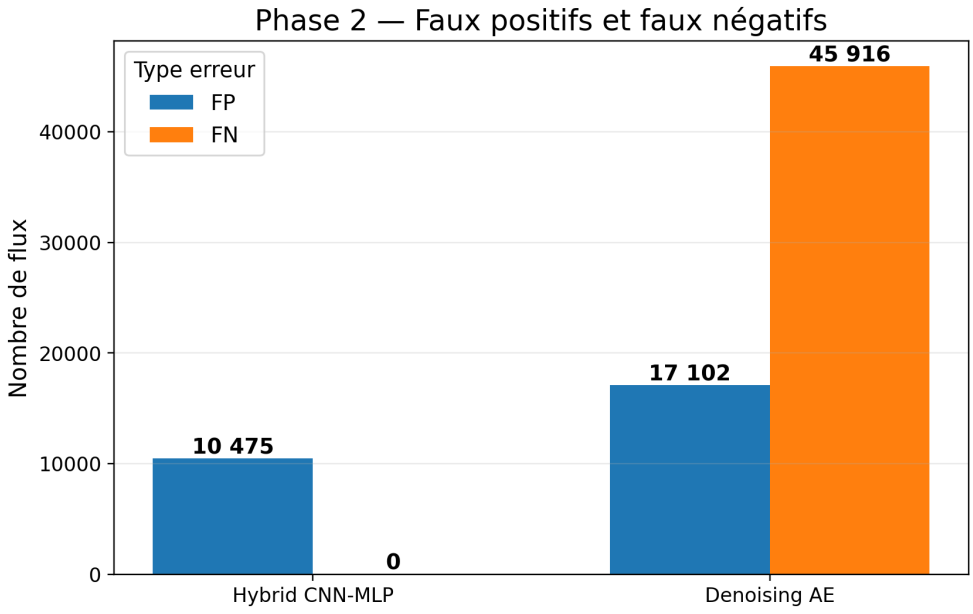

Fig. 4.9 : Comparaison des faux positifs et faux négatifs sur le test global.

4.5.2 Analyse des faux positifs et faux négatifs

La figure 4.9 rend visible la nature différente des erreurs. Le *Hybrid CNN-MLP* transforme essentiellement une faible fraction de BENIGN en alertes. Le *Denoising Autoencoder* produit à la fois plus de FP et surtout beaucoup plus de FN, ce qui est plus problématique pour un IDS. En sécurité réseau, un FP augmente la charge d'analyse, mais un FN laisse une attaque se dérouler sans alerte. C'est donc l'écart sur les FN qui rend la différence entre les deux modèles opérationnellement importante.

4.5.3 Analyse par type d'attaque

La figure 4.10 compare le rappel par famille d'attaque dans le protocole global. Elle complète les métriques binaires : deux modèles peuvent avoir des scores globaux proches tout en se comportant très différemment sur Bot, Web Attack, Infiltration ou SSH-Patator.

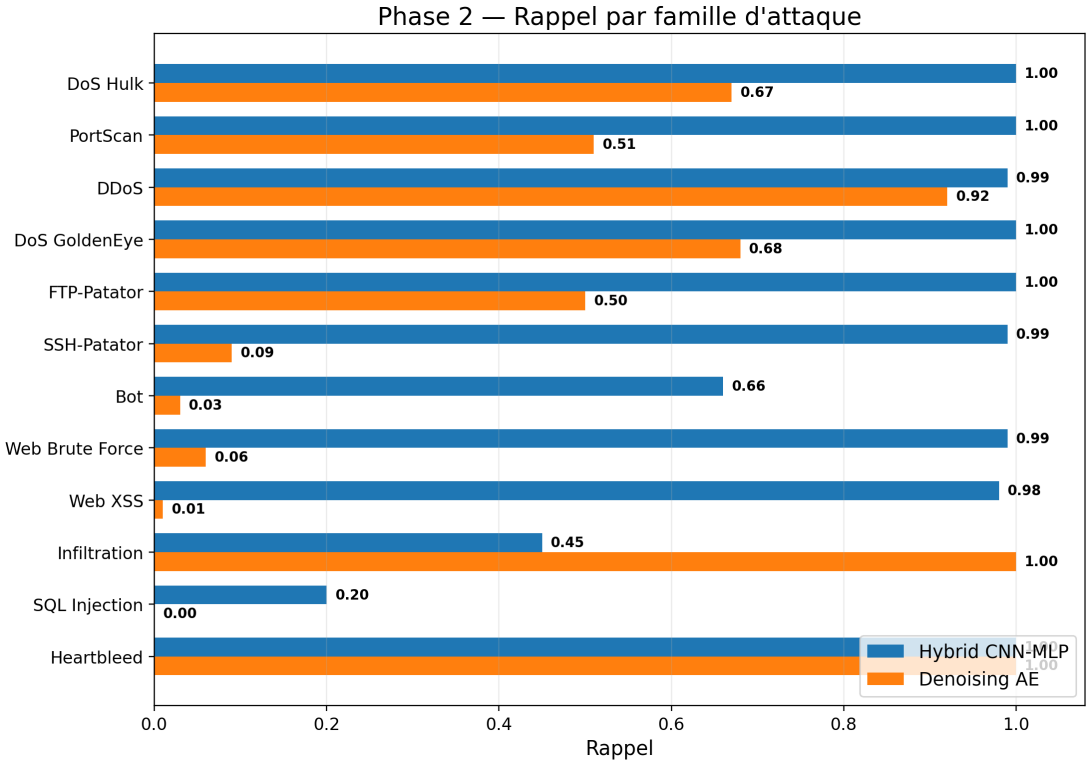

FIG. 4.10 : Rappel par famille d'attaque sur le test global.

Le *Hybrid CNN-MLP* conserve un rappel élevé sur la plupart des familles, ce qui confirme l'intérêt de l'apprentissage supervisé lorsque les labels sont disponibles. L'autoencodeur est plus irrégulier : il réagit bien aux attaques fortement visibles dans les statistiques de flux, comme DDoS ou Heartbleed, mais il échoue davantage sur Bot, SSH-Patator ou certaines attaques web. Cela suggère que ces familles possèdent des signatures numériques moins éloignées du trafic BENIGN après nettoyage et standardisation.

Tab. 4.12 : Lecture qualitative du rappel par famille d'attaque.

Famille	Hybrid CNN-MLP	Denoising Autoencoder
DDoS / DoS	Rappel très élevé ; motifs volumétriques bien appris.	Rappel élevé si l'erreur de reconstruction augmente fortement.
PortScan	Très bonne détection supervisée.	Détection moyenne, sensible au seuil.
FTP/SSH-Patator	Bonne exploitation des relations entre caractéristiques.	Difficultés lorsque les flux ressemblent au trafic normal.
Bot / Web Attack	Meilleure robustesse grâce aux labels.	Rappel faible sur les attaques discrètes ou peu représentées.
Infiltration / Heartbleed	Résultats dépendants du support disponible.	Peut bien réagir si l'anomalie numérique est nette.

Tab. 4.13 : Rappels par famille d'attaque lus à partir de la comparaison globale.

Famille d'attaque	Hybrid CNN-MLP	Denoising AE	Lecture principale
DoS Hulk	1,00	0,67	Avantage net du modèle supervisé.
PortScan	1,00	0,51	L'autoencodeur reste sensible au seuil.
DDoS	0,99	0,92	Les deux modèles détectent bien ce profil volumétrique.
DoS GoldenEye	1,00	0,68	Le modèle hybride conserve un rappel maximal.
FTP-Patator	1,00	0,50	Les labels aident fortement la séparation.
SSH-Patator	0,99	0,09	Profil difficile pour la reconstruction normale.
Bot	0,66	0,03	Famille discrète et faible rappel pour les deux, surtout pour l'AE.
Web Brute Force	0,99	0,06	Forte difficulté de l'AE sur les attaques web.
Web XSS	0,98	0,01	Cas très défavorable à l'AE.
Infiltration	0,45	1,00	Famille atypique : l'AE réagit mieux que le modèle supervisé.
SQL Injection	0,20	0,00	Support rare et détection instable.
Heartbleed	1,00	1,00	Anomalie numérique fortement visible.

Cette analyse par famille d'attaque est importante, car elle montre que la performance d'un IDS ne doit pas être réduite à une seule moyenne globale. Le modèle hybride est meilleur dans cette expérience parce qu'il apprend directement à reconnaître les différents profils d'attaque présents dans l'entraînement. L'autoencodeur, en revanche, dépend de la distance entre chaque attaque et le comportement normal appris. Lorsque cette distance est faible, l'erreur de reconstruction peut rester sous le seuil, ce qui produit des faux négatifs.

4.6 Vérification méthodologique et interprétation

4.6.1 Contrôle du data leakage

Les performances très élevées du *Hybrid CNN-MLP*, notamment $FN = 0$ dans le protocole global, exigent une vérification explicite. Les points de contrôle retenus sont les suivants :

- les colonnes `Label`, `LabelBinary`, `AttackType` et `SourceFile` sont exclues des variables d'entrée ;
- `VarianceThreshold` et `StandardScaler` sont ajustés uniquement sur l'entraînement ;
- le seuil de décision est choisi sur validation, jamais sur le test final ;
- le test global 25 % reste séparé après le split stratifié ;
- Friday DDoS n'est pas utilisé pour ajuster les paramètres de la phase stricte.

Ces précautions n'éliminent pas toutes les limites d'un dataset de laboratoire, mais elles évitent les erreurs les plus courantes d'évaluation, en particulier la normalisation sur l'ensemble complet ou l'utilisation involontaire d'une colonne de label comme variable prédictive.

4.6.2 Interprétation comparative des deux approches

Le *Hybrid CNN-MLP* est plus stable dans cette étude parce qu'il apprend explicitement la frontière entre les deux classes. Sa branche CNN 1D extrait des motifs locaux entre variables voisines dans le vecteur de caractéristiques, tandis que la branche MLP apprend des interactions globales. L'autoencodeur, lui, ne voit pas la classe attaque pendant l'apprentissage principal ; sa performance dépend donc de la distance entre l'attaque et le comportement normal reconstruit.

Tab. 4.14 : Interprétation comparative des deux modèles.

Critère	Denoising Autoencoder	Hybrid CNN-MLP
Nature	Détection d'anomalies centrée sur BENIGN.	Classification supervisée BENIGN/ATTACK.
Score	Erreur de reconstruction.	Probabilité sigmoïde.
Avantage	Utilisable avec peu ou pas d'attaques annotées.	Très performant lorsque les labels sont représentatifs.
Limite	Sensible au seuil et aux attaques proches du normal.	Dépend de la qualité des labels et de la représentativité du train.
Usage conseillé	Module complémentaire pour anomalies inconnues.	Moteur principal de classification IDS hors ligne.

Cette limite ne rend pas l'autoencodeur inutile. Dans un contexte de production, les labels sont rarement aussi propres que dans un dataset de laboratoire. Un autoencodeur entraîné sur du trafic normal récent peut alors servir de module de vigilance pour des comportements nouveaux ou faiblement annotés. Il est donc moins considéré comme un concurrent direct du CNN-MLP que comme un composant complémentaire.

4.6.3 Lecture opérationnelle des résultats

D'un point de vue opérationnel, les résultats peuvent être résumés en trois constats.

Premièrement, en phase stricte, les deux modèles détectent toutes les attaques DDoS du test. Le choix entre eux dépend donc surtout du nombre de fausses alertes produites. Le Hybrid CNN-MLP se montre plus favorable, car il réduit les FP tout en maintenant $FN = 0$.

Deuxièmement, en phase globale, le Hybrid CNN-MLP domine plus nettement. Il obtient un rappel attaque de $1{,}000$ et un F1-score attaque de $0{,}964$, contre respectivement $0{,}670$ et $0{,}747$ pour l'autoencodeur. Cela signifie que, dans ce protocole, la supervision apporte un avantage important lorsque les familles d'attaques du test sont représentées dans les données d'entraînement.

Troisièmement, l'autoencodeur présente une faiblesse sur plusieurs attaques proches du comportement normal. Son score global reste acceptable, mais son nombre élevé de faux négatifs le rend insuffisant comme unique mécanisme de décision dans un IDS critique. Son intérêt réside plutôt dans un rôle complémentaire, par exemple pour surveiller des dérives ou détecter des comportements inhabituels lorsque les labels sont incomplets.

4.6.4 Limites de l'évaluation

Les résultats doivent être interprétés avec prudence. Premièrement, *CICIDS2017* reste un environnement contrôlé et ne couvre pas toute la variabilité d'un réseau de production.

Deuxièmement, certaines familles d'attaques sont peu représentées, ce qui rend leurs scores plus instables. Troisièmement, les expériences sont hors ligne : elles ne mesurent pas la latence, le débit, la consommation mémoire ni l'intégration dans un IDS réel. Enfin, le protocole global utilise un split aléatoire stratifié ; il complète le protocole strict, mais ne remplace pas une validation temporelle sur plusieurs jours futurs.

Une autre limite concerne l'interprétabilité. Le modèle hybride obtient de meilleurs scores, mais il reste nécessaire de comprendre quelles caractéristiques influencent le plus ses décisions. Des méthodes d'explicabilité, comme l'importance des variables, SHAP ou l'analyse des activations, pourraient être ajoutées dans un travail futur afin de rendre le système plus transparent pour un analyste sécurité.

4.7 Discussion et perspectives

La discussion des résultats montre que la performance observée dépend autant du protocole expérimental que du modèle lui-même. Les expérimentations sur *CICIDS2017* restent utiles pour comparer les approches dans un cadre contrôlé, mais les travaux sur les IDS rappellent qu'un réseau réel peut présenter du bruit, une dérive temporelle, des labels imparfaits et des comportements inconnus [5, 21].

La comparaison confirme d'abord que le protocole d'évaluation influence fortement la lecture des performances. Le protocole strict vérifie la capacité du modèle à rester efficace sur une journée séparée, tandis que le protocole global mesure une performance moyenne sur plusieurs familles d'attaques. Ces deux lectures sont complémentaires : une bonne performance globale ne suffit pas à prouver la robustesse temporelle, et un bon résultat sur Friday DDoS ne garantit pas une bonne détection de toutes les attaques.

Sur le plan du positionnement, le *Hybrid CNN-MLP* apparaît comme le meilleur choix lorsque les étiquettes d'apprentissage sont disponibles et représentatives. La branche CNN 1D apporte une lecture locale potentielle des caractéristiques, alors que la branche MLP conserve une lecture globale du flux. Cette combinaison explique sa meilleure stabilité dans les deux protocoles. Toutefois, ce résultat ne doit pas être interprété comme une garantie absolue : les valeurs $FN = 0$ observées dans certains tests décrivent uniquement le découpage expérimental étudié.

Le *Denoising Autoencoder* conserve un intérêt différent. Il est moins performant comme classifieur principal dans les tests supervisés réalisés, mais il reste pertinent comme composant complémentaire lorsque les attaques sont peu annotées ou lorsque l'objectif est de surveiller des écarts par rapport au trafic normal récent. Cette logique est importante en cybersécurité, car les attaques réelles ne sont pas toujours connues à l'avance.

Les perspectives futures se concentrent sur quatre objectifs. Le premier consiste à tes-

ter le pipeline sur d'autres jeux de données, notamment UNSW-NB15, afin de vérifier si les conclusions restent valables hors de *CICIDS2017* [22]. Le deuxième objectif est d'étudier les attaques rares ou non vues pendant l'entraînement, par exemple avec un protocole *Leave-One-Attack-Out*. Le troisième objectif est d'améliorer le traitement du déséquilibre des classes par des pondérations adaptées, des fonctions de perte spécialisées ou des techniques de rééchantillonnage. Le quatrième objectif est opérationnel : mesurer le temps d'inférence, la consommation mémoire et le débit de traitement pour rapprocher le système d'un IDS utilisable sur des flux continus.

Une dernière perspective concerne l'explicabilité. L'ajout de méthodes comme l'analyse d'importance des variables ou SHAP permettrait d'identifier les caractéristiques qui influencent le plus les décisions. Cette étape est nécessaire pour rendre le système plus compréhensible par un analyste sécurité et pour détecter d'éventuelles corrélations artificielles apprises par le modèle.

4.8 Conclusion

Ce chapitre a montré que les résultats ne doivent pas être lus uniquement à travers l'accuracy. La robustesse temporelle, le rappel attaque, les faux positifs et les faux négatifs donnent des informations différentes et complémentaires. En phase stricte, les deux modèles détectent toutes les attaques DDoS, mais le *Hybrid CNN-MLP* produit moins de fausses alertes. En phase globale, l'écart devient plus marqué : le modèle hybride conserve un rappel attaque maximal et un meilleur F1-score, tandis que l'autoencodeur manque un nombre important d'attaques.

Le *Hybrid CNN-MLP* ressort donc comme le modèle le plus efficace lorsque les labels sont disponibles et représentatifs. Le *Denoising Autoencoder* garde néanmoins un intérêt comme module complémentaire pour surveiller des comportements atypiques ou des scénarios où les attaques sont faiblement annotées. Les perspectives retenues concernent surtout la validation sur d'autres jeux de données, l'étude des attaques rares, l'explicabilité et l'évaluation du comportement du système dans des conditions plus proches d'un IDS réel.

Conclusion générale

Cette étude a présenté une chaîne expérimentale complète pour la détection d'anomalies réseau sur le jeu de données *CICIDS2017*. Le travail a couvert la préparation des flux, la prévention du *data leakage*, la standardisation, la calibration des seuils et l'évaluation de deux approches d'apprentissage profond : le *Denoising Autoencoder* et le *Hybrid CNN-MLP*.

L'apport principal est méthodologique. Les résultats d'un IDS ne peuvent pas être jugés uniquement par l'accuracy ; ils doivent être lus à travers le rappel attaque, le F1-score, les faux positifs, les faux négatifs et le protocole utilisé. La séparation stricte entre entraînement, validation, calibration et test final constitue donc une condition essentielle pour éviter des conclusions artificiellement optimistes.

Sur le plan expérimental, le *Hybrid CNN-MLP* obtient les résultats les plus stables dans les deux protocoles étudiés. Sa combinaison d'une lecture locale par CNN 1D et d'une lecture globale par MLP permet de mieux exploiter les caractéristiques numériques des flux lorsque les labels sont disponibles et représentatifs. Le *Denoising Autoencoder* reste moins performant comme moteur principal de décision, mais il garde un intérêt comme module complémentaire de surveillance lorsque les attaques sont peu annotées ou lorsqu'il s'agit de repérer des écarts par rapport au trafic normal.

Ces résultats doivent néanmoins rester limités au cadre expérimental étudié. *CICIDS2017* est un jeu de données contrôlé et ne représente pas toute la complexité d'un réseau réel. Les attaques rares, les distributions changeantes, les labels imparfaits, la latence et la consommation de ressources n'ont pas été entièrement évalués. L'absence de faux négatifs dans certains tests ne constitue donc pas une garantie absolue, mais seulement un résultat observé dans un découpage précis.

Les prolongements naturels consistent à tester le pipeline sur d'autres jeux de données, à étendre l'évaluation au multi-classes, à étudier les attaques non vues pendant l'apprentissage, à intégrer des méthodes d'explicabilité et à mesurer les performances en conditions proches du temps réel. Ces étapes permettraient de rapprocher l'approche d'un IDS plus opérationnel, capable de traiter des flux continus tout en restant interprétable et robuste.

Bibliographie

[1] J. F. Kurose and K. W. Ross, *Computer Networking : A Top-Down Approach*, 8th ed. Pearson, 2021.

[2] A. S. Tanenbaum, N. Feamster and D. J. Wetherall, *Computer Networks*, 6th ed. Pearson, 2021.

[3] W. Stallings, *Network Security Essentials : Applications and Standards*, 6th ed. Pearson, 2017.

[4] K. Scarfone and P. Mell, *Guide to Intrusion Detection and Prevention Systems (IDPS)*, NIST Special Publication 800-94, 2007.

[5] R. Sommer and V. Paxson, ``Outside the Closed World : On Using Machine Learning for Network Intrusion Detection," *IEEE Symposium on Security and Privacy*, pp. 305--316, 2010.

[6] V. Chandola, A. Banerjee and V. Kumar, ``Anomaly Detection : A Survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1--58, 2009.

[7] C. C. Aggarwal, *Outlier Analysis*, 2nd ed. Springer, 2017.

[8] H. He and E. A. Garcia, ``Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263--1284, 2009.

[9] D. M. W. Powers, ``Evaluation : From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37--63, 2011.

[10] T. Saito and M. Rehmsmeier, ``The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets," *PLOS ONE*, vol. 10, no. 3, 2015.

[11] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[12] T. M. Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[13] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016.

[14] Y. LeCun, Y. Bengio and G. Hinton, ``Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436--444, 2015.

[15] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, ``Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278--2324, 1998.

[16] P. Vincent, H. Larochelle, Y. Bengio and P.-A. Manzagol, ``Extracting and Composing Robust Features with Denoising Autoencoders," *ICML*, pp. 1096--1103, 2008.

[17] S. Ioffe and C. Szegedy, ``Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift," *ICML*, pp. 448--456, 2015.

[18] N. Srivastava et al., ``Dropout : A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929--1958, 2014.

[19] I. Sharafaldin, A. H. Lashkari and A. A. Ghorbani, ``Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," *ICISSP*, pp. 108--116, 2018.

[20] Canadian Institute for Cybersecurity, ``Intrusion Detection Evaluation Dataset : CICIDS2017," University of New Brunswick, 2017.

[21] G. Engelen, V. Rimmer and W. Joosen, ``Troubleshooting an Intrusion Detection Dataset : The CICIDS2017 Case Study," *IEEE Security and Privacy Workshops*, pp. 7--12, 2021.

[22] N. Moustafa and J. Slay, ``UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems," *Military Communications and Information Systems Conference*, 2015.

[23] J. Dean and S. Ghemawat, ``MapReduce : Simplified Data Processing on Large Clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107--113, 2008.

[24] M. Zaharia et al., ``Resilient Distributed Datasets : A Fault-Tolerant Abstraction for In-Memory Cluster Computing," *USENIX NSDI*, pp. 15--28, 2012.

[25] T. White, *Hadoop : The Definitive Guide*, 4th ed. O'Reilly Media, 2015.