People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University of Echahid Hamma Lakhdar El Oued

Faculty of Exact Sciences

End-of-Study Dissertation

Academic Master

Field : Computer Science

Specialty : Internet of Things and Network Security

Topic

Network Intrusion Detection Using ML&DL

Presented by : Abdelkarim Habira

Ghezal Nardjes

Publicly defended on : 00-06-2025, before the jury composed of :

Khalaifa abdenacer	**MCA**	**Univ. of El-Oued**	**President**
Dr. Beggas Mounir	**Pr**	**Univ. of El-Oued**	**Supervisor**
Geutas chourouk	**MCA**	**Univ. of El-Oued**	**Examiner**

Academic Year : 2025 – 2026.

Acknowledgements

After thanking God Almighty,
we would like to express our sincere gratitude to everyone who contributed to the completion of this thesis.

We would also like to thank our supervisor, ***Dr.Beggas Mounir*** *, for his constant support and guidance throughout this research.*
His expertise and insight were invaluable in guiding the course of this thesis and essential in helping us improve and develop our research.

We would like to express our sincere gratitude to the members of the examining committee for their time, interest, and valuable efforts in evaluating this work.

Dedication

After years of hard work and continuous effort, years filled with fatigue, patience, and challenges, I stand today on the threshold of graduation to present the fruits of my labor and what I have achieved, by the grace and favor of God. This achievement was not a solitary step, but rather the result of the support, love, and encouragement of people whose presence had the greatest impact on my journey. To those who gave me the strength to reach this point, I dedicate this achievement:

*To the one whose name is inseparable from mine, that great man who taught me the meaning of perseverance and good character, my first supporter, my pillar after God, my pride and joy, to whom I owe all gratitude and appreciation*my father.

To the heart of prayer and the wellspring of tenderness whose door never closed, to the one whose eyes followed my every step, this success of mine is an extension of your prayers and a harvest of the fruits of your love, a humble gift to your great heart. To her, I dedicate my success and my pride, my mother.

*To those whom God strengthened me with, who were the best of helpers,*my sisters and brother.

*To my companion on this journey, who has been and continues to be my supporter in every step, my pillar and partner in My life,*my husband

*To my soulmates and constant companions, I write these words with the absolute certainty that your presence is a source of strength and support for me. Thank you for being in my life .*Malak and Isra

To my soulmates who shared every step of this journey with me, to those who encouraged me to persevere and complete the journey Habira and Yas min.

Today, I dedicate this project as a token of gratitude to God Almighty, who guided me, supported me, and granted me the ability to reach this stage that I have always dreamed of.

Nardjes

Dedication

To those who shaped who I am before I even knew who I wanted to become — to the hands that held mine when the road was steep, the voices that whispered "keep going" when silence felt easier, and the hearts that never once doubted me even when I doubted myself. This work is not merely the result of research and writing; it is the outcome of every sacrifice made around me, every prayer whispered for me, and every moment of love that quietly fuelled my journey. With the deepest gratitude and the sincerest affection, I dedicate these pages:

To the soul that departed and left behind a wound that time cannot heal, yet also left a light that continues to illuminate my path — to the man who planted in me the love of knowledge before I even knew its meaning, who taught me through his life that dignity lies in effort and that nothing is achieved without sacrifice. Though you are no longer here to witness this moment, I feel your presence in every line I have written, and I carry your memory as the greatest honour — ***my beloved father, may God have mercy on his soul.***

To the one who gave without measure and loved without condition, whose prayers were the invisible shield that protected me throughout this journey — to the woman who stayed awake so I could rest, and who carried my worries as if they were her own. Every step I have taken toward this day was fuelled by your endless faith in me. This success is yours before it is mine — ***my dearest mother.***

To those who were my refuge in times of hardship and my companions in moments of joy, who supported me without hesitation and stood beside me without question — ***my brothers and sisters.*** *And to the little ones who brought life and laughter into every difficult day — the children of my brothers and sisters — may this work be a small light on your own path to knowledge someday.*

To my companion on this long road, who shared with me not only the pages of this work but also its burdens, its late nights, and its moments of doubt and hope alike — to the one whose patience and dedication were a constant source of strength for me — ***my dear partner Nardjes.***

To my dearest friends, those rare souls who appeared at just the right moments — with an encouraging word, a reassuring smile, or simply a presence that reminded me I was

not walking this path alone. Your friendship has been one of the most beautiful gifts this journey has given me, and I am grateful for every moment we have shared.

To all of these, and to God who made it all possible, I humbly dedicate this work as a testament to years of effort, patience, and an unwavering belief that the dream was worth every sacrifice.

Habira

Abstract

Network intrusion detection has become a critical challenge in modern cybersecurity, driven by the increasing scale and sophistication of cyberattacks targeting both public and private infrastructure. Traditional machine learning methods, while demonstrating reasonable performance on standardized benchmarks, often struggle to generalize across heterogeneous data sources and fail to detect the complex patterns that distinguish sophisticated attacks from ordinary network traffic. This thesis proposes and evaluates a family of seven multi-class network intrusion detection models built upon a unified experimental framework. The models are trained and tested on a merged dataset combining three publicly available benchmarks — CICIDS2017, CSE-CIC-IDS2018, and a dedicated Network Intrusion Dataset — totaling over 11.8 million network flow records. The study is organized into three main chapters: Chapter 1 establishes the theoretical foundations of network security, key security principles (CIA triad), common attack taxonomies, and IDS architectures including signature-based and anomaly-based detection methods. Chapter 2 reviews the related literature, covering classical machine learning approaches, deep learning-based studies, and benchmark comparisons on standard datasets such as CICIDS2017. Chapter 3 presents the proposed methodology in full detail, including dataset collection and merging, preprocessing pipeline, feature selection (retaining 60 common features), class balancing strategies, model architectures, training procedures, and comprehensive experimental evaluation across multiple datasets and metrics.

The proposed hybrid design philosophy separates deep feature extraction from final classification: a Convolutional Neural Network (CNN) or a Long Short-Term Memory (LSTM) network learns 128-dimensional compressed representations of the input features, whose output vectors are then passed to either a softmax classifier, an XGBoost ensemble, or a Random Forest ensemble. A standalone XGBoost model is also included as a classical reference baseline to isolate the contribution of deep feature extraction.

All seven models — CNN Only, CNN+XGBoost, CNN+Random Forest, LSTM Only, LSTM+XGBoost, LSTM+Random Forest, and XGBoost Only — were trained under identical conditions using the same preprocessing pipeline, data split, and evaluation protocol, enabling direct and unbiased comparison.

The CNN+XGBoost hybrid achieved the best overall balance among all deep learning architectures, attaining an accuracy of 93.96%, an MCC of 0.9273, an AUC of 0.9956, and a detection rate of 96.11% across seven traffic classes. At the per-class level, five of seven classes exceeded 98% DR (DDoS: 99.28%, PortScan: 99.95%, DoS: 98.58%, BruteForce:

98.39%, Botnet: 98.40%), with a Benign DR of 86.05% and a False Alarm Rate of only 3.02%. The model's only persistent weakness was the Infiltration class (DR: 72.41%), reflecting the inherent difficulty in distinguishing this stealthy attack type from normal traffic. In complex multiclass external validation scenarios, CNN+XGBoost demonstrated clear superiority — achieving 87.03% accuracy on the UNSW-NB15 (10-class) dataset versus 83.90% for XGBoost alone, and 98.90% accuracy on the WSN-DS (5-class) dataset with the lowest false alarm rate (0.27%).

The XGBoost Only model, operating directly on the 60 selected features without any deep learning backbone, achieved the highest overall accuracy among all seven models on the merged dataset at 95.29%, an MCC of 0.9435, an AUC of 0.9968, and a detection rate of 97.48%. Per-class results were strong across most categories (DDoS: 99.82%, DoS: 99.90%, PortScan: 99.96%, Botnet: 99.17%, BruteForce: 98.72%), with a Benign DR of 87.23% and an Infiltration DR of 79.38% — the highest Infiltration result among all models. Notably, XGBoost Only demonstrated competitive or superior performance in simpler, well-separated classification tasks and binary classification scenarios, while showing a significant performance gap compared to CNN+XGBoost in complex multiclass settings such as UNSW-NB15.

Across all experiments, CNN-based architectures consistently outperformed LSTM-based ones on tabular network flow data, and hybrid models consistently outperformed their standalone deep learning counterparts. These findings confirm that the optimal model selection must account for dataset complexity, class count, and feature separability — and that combining deep representation learning with ensemble classification methods provides a strong and scalable foundation for building accurate, generalizable, and intelligent intrusion detection systems.

Keywords: Intrusion Detection System, Deep Learning, CNN, LSTM, XGBoost, Random Forest, Hybrid Model, CICIDS2017, CSE-CIC-IDS2018, UNSW-NB15, WSN-DS, Network Security.

Résumé

La détection d'intrusions réseau est devenue un défi critique en matière de cybersécurité moderne, en raison de l'ampleur et de la sophistication croissantes des cyberattaques ciblant les infrastructures publiques et privées. Les méthodes classiques d'apprentissage automatique, bien qu'offrant des performances acceptables sur des benchmarks standardisés, peinent souvent à généraliser sur des sources de données hétérogènes et échouent à détecter les schémas complexes qui distinguent les attaques sophistiquées du trafic réseau ordinaire.

Ce mémoire propose et évalue une famille de sept modèles de détection d'intrusions réseau multi-classes, construits sur un cadre expérimental unifié. Les modèles sont entraînés et testés sur un jeu de données fusionné combinant trois benchmarks publics — CICIDS2017, CSE-CIC-IDS2018 et un jeu de données dédié aux intrusions réseau — totalisant plus de 11,8 millions d'enregistrements de flux réseau. L'étude est organisée en trois chapitres principaux : le Chapitre 1 établit les fondements théoriques de la sécurité réseau, les principes clés de sécurité (triade CIA), les taxonomies d'attaques courantes, ainsi que les architectures IDS incluant les méthodes de détection par signature et par anomalie. Le Chapitre 2 passe en revue la littérature connexe, couvrant les approches classiques d'apprentissage automatique, les études basées sur l'apprentissage profond, et les comparaisons sur des jeux de données standards tels que CICIDS2017. Le Chapitre 3 présente en détail la méthodologie proposée, incluant la collecte et la fusion des données, le pipeline de prétraitement, la sélection de caractéristiques (60 features communes retenues), les stratégies d'équilibrage des classes, les architectures des modèles, les procédures d'entraînement, et une évaluation expérimentale complète sur plusieurs jeux de données et métriques.

La philosophie de conception hybride proposée sépare l'extraction de caractéristiques profondes de la classification finale : un réseau de neurones convolutif (CNN) ou un réseau à mémoire à long terme (LSTM) apprend des représentations compressées de dimension 128 à partir des features d'entrée, dont les vecteurs de sortie sont ensuite transmis à un classificateur softmax, à un ensemble XGBoost, ou à un ensemble Random Forest. Un modèle XGBoost autonome est également inclus comme référence classique afin d'isoler la contribution de l'extraction de caractéristiques profondes.

Les sept modèles — CNN Only, CNN+XGBoost, CNN+Random Forest, LSTM Only, LSTM+XGBoost, LSTM+Random Forest et XGBoost Only — ont été entraînés dans des conditions identiques, avec le même pipeline de prétraitement, la même segmentation

des données et le même protocole d'évaluation, permettant une comparaison directe et objective. Le modèle hybride CNN+XGBoost a obtenu le meilleur équilibre global parmi toutes les architectures d'apprentissage profond, avec une précision de 93,96%, un MCC de 0,9273, une AUC de 0,9956 et un taux de détection de 96,11% sur sept classes de trafic. Au niveau par classe, cinq classes sur sept ont dépassé 98% de DR (DDoS : 99,28%, PortScan : 99,95%, DoS : 98,58%, BruteForce : 98,39%, Botnet : 98,40%), avec un DR Bénin de 86,05% et un taux de fausses alarmes de seulement 3,02%. La seule faiblesse persistante du modèle concerne la classe Infiltration (DR : 72,41%), reflétant la difficulté inhérente à distinguer ce type d'attaque furtive du trafic normal. Dans des scénarios de validation externe multiclasses complexes, CNN+XGBoost a démontré une nette supériorité — atteignant 87,03% de précision sur le jeu de données UNSW-NB15 (10 classes) contre 83,90% pour XGBoost seul, et 98,90% de précision sur WSN-DS (5 classes) avec le taux de fausses alarmes le plus bas (0,27%).

Le modèle XGBoost Only, opérant directement sur les 60 features sélectionnées sans aucun backbone d'apprentissage profond, a obtenu la précision globale la plus élevée parmi les sept modèles sur le jeu de données fusionné, avec 95,29%, un MCC de 0,9435, une AUC de 0,9968 et un taux de détection de 97,48%. Les résultats par classe sont excellents dans la majorité des catégories (DDoS : 99,82%, DoS : 99,90%, PortScan : 99,96%, Botnet : 99,17%, BruteForce : 98,72%), avec un DR Bénin de 87,23% et un DR Infiltration de 79,38% — le meilleur résultat Infiltration parmi tous les modèles. XGBoost Only a démontré des performances compétitives voire supérieures dans les tâches de classification simples et bien séparées ainsi que dans les scénarios de classification binaire, tout en accusant un écart significatif face à CNN+XGBoost dans les contextes multiclasses complexes tels que UNSW-NB15.

Sur l'ensemble des expériences, les architectures basées sur CNN ont systématiquement surpassé celles basées sur LSTM sur des données tabulaires de flux réseau, et les modèles hybrides ont constamment surpassé leurs homologues d'apprentissage profond autonomes. Ces résultats confirment que le choix optimal du modèle doit tenir compte de la complexité du jeu de données, du nombre de classes et de la séparabilité des features — et que la combinaison de l'apprentissage de représentations profondes avec des méthodes de classification par ensemble constitue une base solide et scalable pour construire des systèmes de détection d'intrusion précis, généralisables et intelligents.

Mots-clés : Système de Détection d'Intrusion, Apprentissage Profond, CNN, LSTM, XGBoost, Random Forest, Modèle Hybride, CICIDS2017, CSE-CIC-IDS2018, UNSW-NB15, WSN-DS, Sécurité Réseau.

ملخص

أصبح الكشف عن اختراقات الشبكات تحدياً بالغ الأهمية في مجال الأمن السيبراني الحديث، مدفوعاً بالتزايد المستمر في حجم الهجمات الإلكترونية وتطورها، والتي تستهدف البنى التحتية العامة والخاصة على حدٍّ سواء. تُقترح في هذه الأطروحة وتُقيَّم مجموعة من سبعة نماذج متعددة الفئات للكشف عن اختراقات الشبكات، مبنية على إطار تجريبي موحد. تم تدريب هذه النماذج واختبارها على مجموعة بيانات مدمجة تجمع بين ثلاث قواعد بيانات متاحة للعموم — CICIDS2017 وCSE-CIC-IDS2018 ومجموعة بيانات اختراق الشبكات المخصصة — بإجمالي يتجاوز 8.11 مليون سجل تدفق شبكي.

يتضمن البحث ثلاثة فصول رئيسية: يرسي الفصل الأول الأسس النظرية لأمن الشبكات، ومبادئ الأمان الأساسية (مثلث CIA)، وتصنيفات الهجمات الشائعة، وبنيات أنظمة الكشف عن الاختراقات بما فيها أساليب الكشف المعتمدة على التوقيعات والكشف القائم على الشذوذ. ويستعرض الفصل الثاني الأدبيات ذات الصلة، شاملاً مناهج التعلم الآلي الكلاسيكية، والدراسات القائمة على التعلم العميق، والمقارنات المرجعية على مجموعات البيانات القياسية كـ CICIDS2017. أما الفصل الثالث فيقدم المنهجية المقترحة بالتفصيل الكامل، بما في ذلك جمع البيانات ودمجها، وخط أنابيب المعالجة المسبقة، واختيار الميزات مع الإبقاء على 60 ميزة مشتركة، واستراتيجيات موازنة الفئات، وبنيات النماذج، وإجراءات التدريب، والتقييم التجريبي الشامل عبر مجموعات بيانات ومقاييس متعددة.

تعتمد فلسفة التصميم الهجين المقترحة على الفصل بين استخراج الميزات العميقة والتصنيف النهائي: إذ تتعلم شبكة عصبية تلافيفية (CNN) أو شبكة ذاكرة طويلة-قصيرة المدى (LSTM) تمثيلات مضغوطة ذات 128 بُعداً للميزات المدخلة، ثم تُمرَّر متجهات المخرجات إما إلى مصنف softmax، أو مجموعة XGBoost، أو مجموعة Random Forest. كما يُدرج نموذج XGBoost منفرد بوصفه خطاً أساسياً مرجعياً كلاسيكياً لعزل مساهمة استخراج الميزات العميقة.

جميع النماذج السبعة — Only CNN وCNN+XGBoost وCNN+Random Forest وOnly LSTM وLSTM+XGBoost وForest LSTM+Random وOnly XGBoost — دُرِّبت في ظروف متطابقة باستخدام نفس خط أنابيب المعالجة المسبقة، وتقسيم البيانات، وبروتوكول التقييم، مما يتيح مقارنة مباشرة وغير متحيزة.

حقق النموذج الهجين CNN+XGBoost أفضل توازن شامل بين جميع بنيات التعلم العميق، إذ بلغت دقته 96.93%، ومعامل ماثيوس للارتباط 0.9273، ومساحة تحت المنحنى 0.9956، ومعدل كشف 11.96% عبر سبع فئات من حركة المرور. على مستوى الفئات، تجاوزت خمس فئات من أصل سبع نسبة 98% في معدل الكشف (DDoS: 99.28%، PortScan: 99.95%، DoS: 98.58%، BruteForce: 98.39%، Botnet: 98.40%)، مع معدل كشف للفئة الحميدة بلغ 86.05% ومعدل إنذار كاذب لم يتجاوز 3.02%. وكانت نقطة الضعف الوحيدة المستمرة للنموذج هي فئة الاختراق (معدل الكشف: 72.41%)، مما يعكس الصعوبة الجوهرية في تمييز هذا النوع الخفي من الهجمات عن حركة المرور الطبيعية. في سيناريوهات التحقق الخارجي متعددة الفئات المعقدة، أثبت CNN+XGBoost تفوقاً واضحاً، إذ حقق دقة 87.03% على مجموعة بيانات UNSW-NB15 ذات 10 فئات مقارنةً بـ 83.90% لنموذج XGBoost منفرداً، ودقة 98.90% على مجموعة بيانات WSN-DS ذات 5 فئات مع أدنى معدل إنذار كاذب بلغ 0.27%.

أما نموذج XGBoost، Only العامل مباشرةً على الـ 60 ميزة المختارة دون أي عمود فقري للتعلم العميق، فقد حقق أعلى دقة إجمالية بين النماذج السبعة على مجموعة البيانات المدمجة بنسبة 95.29%، ومعامل ماثيوس 0.9435، ومساحة تحت المنحنى 0.9968، ومعدل كشف 97.48%. وكانت نتائج الفئات قوية في معظم الفئات (DDoS: 99.82%، DoS:

72%.98 BruteForce: ،17%.99 Botnet: ،96%.99 PortScan: ،90%.99)، مع معدل كشف للفئة الحميدة بلغ 23%.87 وللاختراق 38%.79 — وهو أعلى نتيجة لفئة الاختراق بين جميع النماذج. والجدير بالذكر أن XGBoost Only أظهر أداءً تنافسياً أو متفوقاً في مهام التصنيف البسيطة جيدة الفصل والتصنيف الثنائي، في حين أظهر قوة أداء ملحوظة مقارنةً بـ CNN+XGBoost في البيئات متعددة الفئات المعقدة كـ UNSW-NB15.

عبر جميع التجارب، تفوقت البنيات القائمة على CNN باستمرار على تلك القائمة على LSTM في بيانات تدفق الشبكات الجدولية، وتفوقت النماذج الهجينة باستمرار على نظيراتها المنفردة من التعلم العميق. تؤكد هذه النتائج أن اختيار النموذج الأمثل يجب أن يراعي تعقيد مجموعة البيانات، وعدد الفئات، وقابلية فصل الميزات — وأن الجمع بين تعلم التمثيل العميق وأساليب التصنيف التجميعي يوفر أساساً قوياً وقابلاً للتوسع لبناء أنظمة كشف عن الاختراقات دقيقة وقابلة للتعميم وذكية.

الكلمات المفتاحية: نظام الكشف عن الاختراقات، التعلم العميق، CNN، LSTM، XGBoost Random، Forest النموذج الهجين، CICIDS2017، CSE-CIC-IDS2018، UNSW-NB15، WSN-DS، أمن الشبكات.

Contents

List of Figures **xiv**

List of Tables **xvi**

List of Abbreviations **xvii**

General Introduction **1**

Chapter 1 Network Security & Intrusion Detection System **3**

1.1 Introduction 4
1.2 Key Pillars of Network Security 4
1.2.1 Confidentiality 5
1.2.2 Integrity 5
1.2.3 Availability 5
1.3 Threats to Network Security 6
1.3.1 Denial-of-Service (DoS) 6
1.3.2 Distributed denial-of-service (DDoS) 7
1.3.3 Brute force 8
1.3.4 SQL injection 10
1.3.5 Infiltration 10
1.3.6 Botnet 11
1.4 Network Security Measures 11
1.4.1 Firewalls 11
1.4.2 Virtual Private Networks (VPNs) 13
1.5 Intrusion Detection System(IDS) 13
1.6 Types of IDS 13
1.6.1 Network-based IDS 13
1.6.2 Host-based IDS 14
1.7 IDS Architecture 15
1.7.1 Data Collection 16
1.7.2 Data pre-processor 16

1.7.3 Intrusion recognition 16
1.8 conclusion 16

Chapter 2 Background & Related Work 18
2.1 Introduction 19
2.2 Traditional Intrusion Detection Approaches 19
2.2.1 Anomaly-Based Detection 19
2.2.2 Signature-Based detection 22
2.2.3 Hybrid IDS 24
2.3 Machine Learning Approaches 24
2.4 Deep Learning Approaches 26
2.5 Conclusion 27

Chapter 3 Building and Evaluating Machine Learning and Deep Learning Models for IDS 28
3.1 Introduction 29
3.2 Execution environment 29
3.3 **Evaluating the proposed models on merging CIC-IDS2017 and CSE-CIC-IDS2018 and Network Intrusion Dataset datasets** 32
3.3.1 Data Sets 32
3.3.2 Overview of the work pipeline 35
3.3.3 Data Preprocessing 36
3.3.4 **Description and Architecture of the Proposed Models** . . . 43
3.3.5 **Training and Validation** 47
3.3.6 **Results and Analysis** 54
3.3.7 Comparative Analysis 62
3.4 **Per-Dataset Comparative Evaluation of the Proposed Models and State-of-the-Art Models** 64
3.4.1 Analyzing feature indicators on different datasets 64
3.4.2 Analysis of the results of the comparisons 64
3.4.3 The Advantage of CNN+XGBoost in Complex Multiclass Scenarios 81
3.5 Conclusion 82

List of Figures

1.1 CIA triad. 4
1.2 DoS attack 6
1.3 DDoS attack 7
1.4 Brute force attack 9
1.5 SQL injection 10
1.6 Firewall. 12
1.7 Network-based IDS. 14
1.8 Host-based IDS 15
1.9 Framework of Intrusion Detection System 15

3.1 Component architecture of the proposed work 35
3.2 Training and Validation CNN + XGBoost 47
3.3 Training and Validation Loss CNN + XGBoost 48
3.4 Confusion Matrices CNN + XGBoost 50
3.5 Training and Validation XGBoost 51
3.6 Training and Validation Loss XGBoost 52
3.7 Confusion Matrices XGBoost 53
3.8 Confusion Matrices CNN+XGBoost 54
3.9 Training and Validation Loss and Accuracy Curves of the CNN Backbone for the CNN+XGBoost Model 65
3.10 XGBoost Only 66
3.11 CNN+XGBoost 66
3.12 Confusion Matrices – CIC-IDS2017 Binary Classification 66
3.13 Training and Validation Loss and Accuracy Curves of the CNN Backbone for the CNN+XGBoost Model 68
3.14 XGBoost Only 69
3.15 CNN+XGBoost 69
3.16 Confusion Matrices – CIC-IDS2017 multiclass Classification 69
3.17 Training and Validation Loss and Accuracy Curves of the CNN-XGBoost Only Model 71

3.18 Training and Validation Loss and Accuracy Curves of the XGBoost Only Model 72
3.19 Confusion Matrices — Binary Classification on UNSW-NB15 (CNN+XGBoost and XGBoost Only 73
3.20 Training and Validation Loss and Accuracy Curves of the CNN Backbone for the CNN+XGBoost Model 74
3.21 Confusion Matrix — CNN+XGBoost 75
3.22 Confusion Matrix — XGBoost 75
3.23 Training and Validation Loss and Accuracy Curves of the CNN+XGBoost Model 77
3.24 Confusion Matrix — XGBoost 78
3.25 Confusion Matrix — CNN+XGBoost (WSN-DS Binary) 78
3.26 Training and Validation Accuracy and Loss Curves of the CNN Backbone for the CNN+XGBoost Model (WSN-DS — Multiclass Classification, 5 Classes) 80

List of Tables

2.1 Performance of ML models on UNSW-NB15 – Ajagbe et al. (2024) [15] . . 25
2.2 Performance of classical ML models on CICIDS2017 -Obagbuwa et al.(2026)[16]. 25
2.3 Performance of Deep Learning model on CICIDS2017 – Qazi et al. (2022)[17]. 26
2.4 The comparison among LSTM model and another method 27

3.1 List of python libraries 31
3.2 CICIDS2017 Files after Label Normalization 33
3.3 Description of files containing the CIC-IDS2018 dataset 34
3.4 Label Unification Mapping Across Datasets 38
3.5 Class Distribution After Downsampling 39
3.6 Train/Test Split 40
3.7 Label Encoding Mapping 41
3.8 CNN Only — Layer Configuration 44
3.9 LSTM Only — Layer Configuration 45
3.10 Summary of All Model Configurations 47
3.11 LSTM — Overall Performance Metrics 55
3.12 LSTM Only — Per-Class Results 55
3.13 CNN Only — Overall Performance Metrics 56
3.14 CNN Only — Per-Class Results 56
3.15 LSTM + Random Forest — Overall Performance Metrics 57
3.16 LSTM + Random Forest — Per-Class Results 57
3.17 CNN + Random Forest — Overall Performance Metrics 58
3.18 CNN + Random Forest — Per-Class Results 58
3.19 LSTM + XGBoost — Overall Performance Metrics 59
3.20 LSTM + XGBoost — Per-Class Results 59
3.21 CNN + XGBoost — Overall Performance Metrics 60
3.22 CNN + XGBoost — Per-Class Results 60
3.23 XGBoost Only — Overall Performance Metrics 61
3.24 XGBoost Only — Per-Class Results 61
3.25 Comparative Performance of All Seven Models (Worst to Best) 62
3.26 Detection Rate (%) per Class Across All Seven Models 62

3.27 Final Model Ranking and Key Strengths 63
3.28 CIC-IDS2017 — Binary Classification (%) 64
3.29 CIC-IDS2017 — Multiclass Classification, 7 Classes (%) 67
3.30 UNSW-NB15 — Binary Classification (%) 70
3.31 UNSW-NB15 — Multiclass Classification, 10 Classes (%) 73
3.32 WSN-DS — Binary Classification (%) 76
3.33 WSN-DS — Multiclass Classification, 5 Classes (%) 79

List of Abbreviations

Abbreviation	Full Term
ACC	Accuracy
ANN	Artificial Neural Network
AUC	Area Under the ROC Curve
AWS	Amazon Web Services
CIA	Confidentiality, Integrity, Availability
CIC	Canadian Institute for Cybersecurity
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSE	Communications Security Establishment
CSV	Comma-Separated Values
DDoS	Distributed Denial of Service
DL	Deep Learning
DoS	Denial of Service
DR	Detection Rate
F1	F1-Score
FAR	False Alarm Rate
FTP	File Transfer Protocol
GPU	Graphics Processing Unit
HDD	Hard Disk Drive
HIDS	Host-based Intrusion Detection System
HMM	Hidden Markov Model
HOIC	High Orbit Ion Cannon
HTTP	Hypertext Transfer Protocol
HULK	HTTP Unbearable Load King
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol

Abbreviation	Full Term
IPS	Intrusion Prevention System
LOIC	Low Orbit Ion Cannon
LSTM	Long Short-Term Memory
MCC	Matthews Correlation Coefficient
ML	Machine Learning
NaN	Not a Number
NGFW	Next-Generation Firewall
NIDS	Network-based Intrusion Detection System
PCAP	Packet Capture
PR	Precision
RAM	Random Access Memory
RF	Random Forest
ROC	Receiver Operating Characteristic
SMOTE	Synthetic Minority Oversampling Technique
SQL	Structured Query Language
SSH	Secure Shell
SSD	Solid State Drive
TCP	Transmission Control Protocol
TPU	Tensor Processing Unit
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VPN	Virtual Private Network
XGBoost	Extreme Gradient Boosting

General Introduction

The rapid expansion of computer networks and the growing dependence on digital infrastructure have made cybersecurity one of the most critical concerns of the modern era. Organizations across all sectors ,from government institutions to private enterprises rely heavily on networked systems to store, process, and transmit sensitive information. This increasing connectivity, while enabling unprecedented levels of productivity and communication, has simultaneously exposed these systems to a wide and ever-evolving spectrum of cyber threats.

Network intrusion detection has emerged as a fundamental line of defense against such threats. An Intrusion Detection System (IDS) monitors network traffic in real time, identifying suspicious activities and potential security breaches before they can cause irreversible damage. Traditional IDS approaches, whether based on predefined attack signatures or statistical anomaly models, have long served as the backbone of network security. However, as cyberattacks grow in sophistication, volume, and diversity, these conventional methods increasingly struggle to keep pace. Signature-based systems fail to detect novel or zero-day attacks, while anomaly-based systems often produce excessive false alarms, both of which undermine the reliability and practicality of intrusion detection in real-world environments.

The emergence of machine learning and deep learning has opened new horizons for intrusion detection research. Unlike rule-based systems, machine learning models can automatically learn discriminative patterns from large volumes of network traffic data, adapting to new attack behaviors without requiring manual signature updates. Deep learning architectures, in particular, have demonstrated a remarkable ability to extract complex hierarchical representations from raw data, making them well-suited for detecting subtle and sophisticated attack patterns that traditional methods fail to capture.

Despite these advances, several challenges remain. Most existing studies evaluate their models on a single benchmark dataset, limiting the generalizability of their conclusions. Furthermore, many deep learning approaches rely on end-to-end classification

architectures whose softmax output layers struggle to form sharp decision boundaries between traffic classes with overlapping statistical characteristics, such as Infiltration versus Benign traffic, or DDoS versus DoS attacks. The question of whether deep feature extraction alone is sufficient, or whether combining it with powerful ensemble classifiers can yield further improvements, has not been thoroughly investigated across diverse and heterogeneous data sources.

This thesis addresses these gaps by proposing and evaluating a family of hybrid deep learning architectures for multi-class network intrusion detection. The study combines three publicly available benchmark datasets CICIDS2017, CSE-CIC-IDS2018, and the Network Intrusion Dataset into a unified training and evaluation framework comprising over 11.8 million network flow records. Six hybrid models are proposed, each pairing a deep learning backbone — either a Convolutional Neural Network (CNN) or a Long Short-Term Memory (LSTM) network with an ensemble classifier, either XGBoost or Random Forest, for the final classification decision. A standalone XGBoost model is also included as a classical reference baseline to isolate and quantify the contribution of deep feature extraction.

The work is organized into three chapters .

Chapter 1 presents the fundamental concepts of network security and intrusion detection systems, covering the key security principles, the most common attack types, and the main IDS architectures and detection methods

Chapter 2 reviews the theoretical background and the most relevant related work, examining traditional detection approaches, classical machine learning methods, and recent deep learning-based studies in the context of network intrusion detection.

Chapter 3 describes the proposed methodology in full detail, including the dataset collection and merging process, the preprocessing pipeline, the architecture and training of all seven models, and a comprehensive analysis of the experimental results across multiple evaluation metrics and datasets.

Chapter 1

Network Security & Intrusion Detection System

1.1 Introduction

Network security is the process of developing and implementing a defensive strat egy to safeguard the underlying networking infrastructure from unauthorized access, malicious activity, potential threats, or intrusion attempts. It is a critical aspect of any organization's operations, as it ensures the confidentiality, integrity, and avail ability of data and systems. Intrusion Detection Systems (IDS) are monitoring systems that listen to network traffic in a stealthy manner to detect potential se curity breaches. They are considered essential components of network security, as they help identify and respond to potential threats in real-time, mitigating the risk of security breaches. IDS can detect both known and unknown threats, making them an indispensable tool for protecting against emerging cyber-attacks.

1.2 Key Pillars of Network Security

Information security relies on a set of fundamental principles that guide how organizations protect their valuable assets. At its core is the CIA triad — Confidentiality, Integrity, and Availability — a model that has long served as the foundation of information security practices[1].

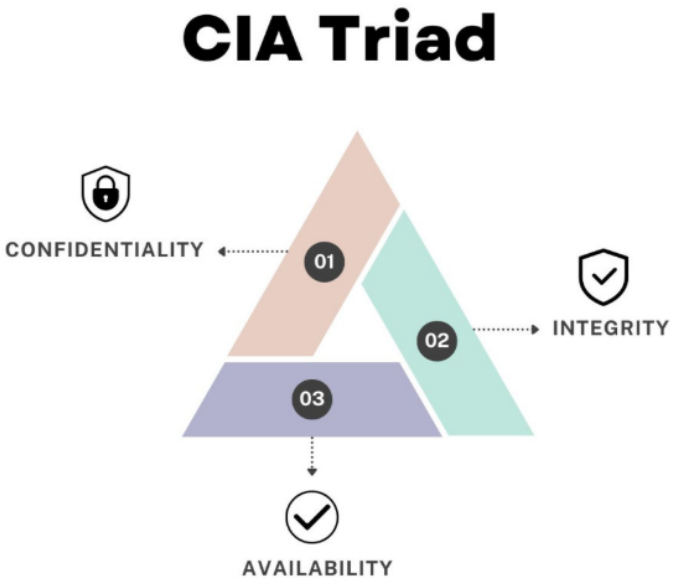


Figure 1.1: CIA triad.

1.2.1 Confidentiality

Confidentiality means keeping data a secret from everyone except those who we want to access it. The ISO/IEC 27000:2018 defines it as "The property that information is not made available or disclosed to unauthorized individuals, entities, or processes." In practice, confidentiality ensures that sensitive information is protected from unauthorized access or disclosure. This is often achieved through mechanisms like encryption, access controls, and secure communication channels.

To apply confidentiality in real environments, you also need to understand how it is enforced across the different layers of network security so that controls at the physical, network, and application layers all work together instead of leaving gaps attackers can exploit.

1.2.2 Integrity

Integrity ensures that data hasn't become corrupted, tampered with, or altered in an unauthorized manner. The ISO/IEC 27000:2018 defines it as "The property of accuracy and completeness." Maintaining data integrity means ensuring that information remains accurate, consistent, and trustworthy throughout its lifecycle. This is often achieved through methods such as hashing, digital signatures, and version control systems.

1.2.3 Availability

Availability means that data is readily accessible to authorized parties when they need it. The ISO/IEC 27000:2018 defines it as "The property of being accessible and usable upon demand by an authorized entity." Ensuring availability involves implementing robust systems, backup solutions, and disaster recovery plans to prevent disruptions and maintain access to critical information and services.

Organizations must also consider legal and geopolitical factors like data sovereignty and data residency when designing secure, compliant systems that maintain data availability across regions.

Apart from the CIA triad, there is another set of measures that should be put in place to ensure the security of information. These measures are known as authenti cation, authorization, and accounting.

1.3 Threats to Network Security

Networks face various types of security threats that can compromise the security and integrity of our data such as viruses, malware, and hacking. These threats can cause serious damage to the network, resulting in data loss, system downtime, and financial losses. We cite the most common network attacks below.

1.3.1 Denial-of-Service (DoS)

A denial-of-service (DoS) attack is a type of cyberattack in which the attacker seeks to disrupt a system and make it unavailable to legitimate users by preventing it from functioning normally. DoS attacks typically work by flooding the targeted system with a large number of requests, preventing it from processing normal traffic and thereby denying users access to the service. A DoS attack is characterized by the fact that it relies solely on a single device to launch the attack[2].



Figure 1.2: DoS attack

1.3.1.1 Hulk

HULK (HTTP Unbearable Load King) is a class of application-layer (Layer 7) HTTP flood attack that deliberately generates large volumes of unique, cache-busting requests to overwhelm web servers and application stacks. Originally released as a research/stress-testing tool in 2012, HULK's technique—randomizing query strings, headers and URIs to defeat caching and signature detection—has been adopted and adapted by attackers to build scalable, hard-to-detect HTTP floods. Because these attacks mimic legitimate HTTP behavior at scale, defending against HULK-style floods requires behavior-based detection, adaptive rate control and tight integration between application security and network defenses.

1.3.1.2 GoldenEye

GoldenEye is a free tool available to everyone on GitHub, designed to carry out denial-of-service attacks. This tool was built using the .NET Core framework. It also includes many essential components and plugins that can be relied upon for daily work. This tool allows a single device to take down another device's web server by using HTTP requests that appear normal. It establishes a full TCP connection, then requires only a limited number of requests sent at regular intervals over a relatively long period. As a result, the tool does not need to generate a large amount of traffic to consume the server's available bandwidth[3].

1.3.2 Distributed denial-of-service (DDoS)

A distributed denial-of-service (DDoS) attack is a malicious attack designed to disrupt the normal operation of a targeted server, service, or network system by flooding the surrounding infrastructure with a massive volume of Internet traffic.

Distributed denial-of-service (DDoS) attacks increase their effectiveness by relying on a network of compromised computer systems as launch points for the attack. The devices used can include computers and network-connected resources, such as Internet of Things (IoT) devices.

In general, a DDoS attack can be likened to a sudden traffic jam that blocks a highway, preventing the normal flow of data from reaching its destination[4].

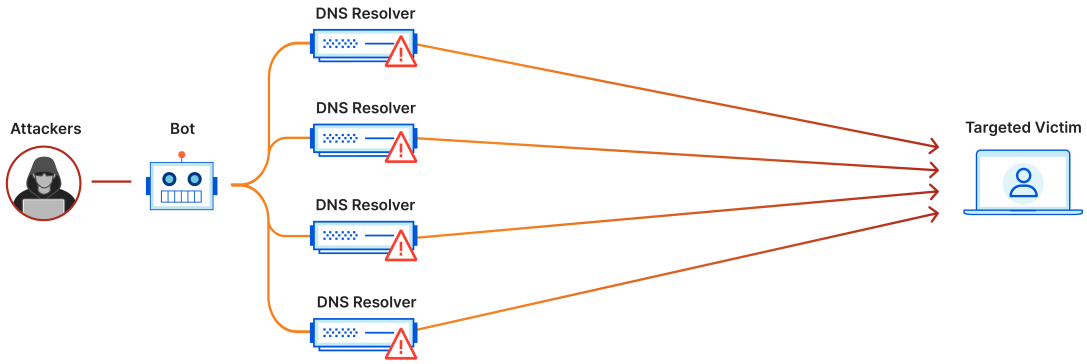


Figure 1.3: DDoS attack

1.3.2.1 HOIC

The High Orbit Ion Cannon is one of the most well-known tools used to carry out denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks, as it floods the victim's network with a large volume of web requests in order to take down a web service or a specific service. This tool is an easily accessible open-source tool created by the group Anonymous, and it is an upgraded version known as the Low Orbit Ion Cannon. The names of both tools were inspired by weapons found in science fiction games. While most software tools require a high level of technical skill, HOIC offers a user-friendly interface and can be easily launched with a single click.

Although it is used in many malicious and illegal attacks, the HOIC is still legally available because it has applications as a legitimate testing tool for users who want to implement a "stress test" on their own networks.[5]

1.3.2.2 LOIC

The Low Orbit Ion Cannon is one of the most well-known open-source tools used for network stress testing and launching denial-of-service attacks. Attackers can use this tool to overwhelm networks and websites with heavy traffic; it is often used in coordinated distributed denial-of-service (DDoS) attacks.

LOIC performs a DoS attack by flooding the target with TCP, UDP, or HTTP requests. This can overwhelm the server's resources like CPU, memory, and bandwidth resulting in slow response or unavailability for legitimate users.[6]

1.3.3 Brute force

Brute-force attacks are among the most common hacking methods, and some studies indicate that this type of attack accounts for a significant portion of confirmed security breaches. A brute-force attack involves trying different usernames and passwords to gain unauthorized access to a system. This method is considered one of the simplest attack techniques and has a high success rate.

Some attackers rely on applications and scripts as tools for launching brute-force attacks, in which these tools test numerous different passwords in an attempt to bypass authentication processes. In other cases, attackers attempt to compromise web applications by trying to find the correct session ID. Attackers' motives may include stealing

data, infecting websites with malware, or disrupting services.

While some attackers still perform brute force attacks manually, today almost all brute force attacks today are performed by bots. Attackers have lists of commonly used credentials, or real user credentials, obtained via security breaches or the dark web. Bots systematically attack websites and try these lists of credentials, and notify the attacker when they gain access[7][8].

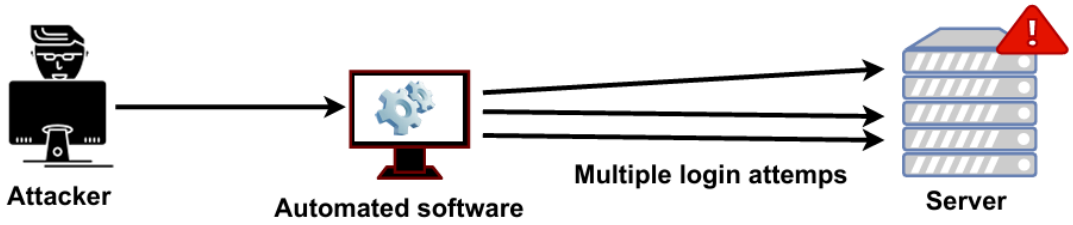


Figure 1.4: Brute force attack

Four different types of brute force attacks are listed below:

1.3.3.1 FTP Brute Force

The File Transfer Protocol (FTP) is a network communication protocol used for transferring files. It employs a client-server architecture, whereby users can connect to the server using an FTP client. Authentication is performed using login credentials, which are typically sent unencrypted; anonymous login may also be supported if available.

1.3.3.2 SSH Brute Force

This type of attack targets the Secure Shell (SSH) protocol, which is used to remotely access and manage servers. The attacker tries various combinations of usernames and passwords until they successfully log in to the SSH server. Once they gain access, they can carry out various malicious activities, such as stealing data or planting malware.

1.3.3.3 Web Brute Force

This attack targets electronic authentication portals, such as those used for email or online banking services. The attacker uses automated software to try various usernames and passwords until they gain access to the system. Once they have control of it, they can steal information or carry out illegal activities.

1.3.3.4 XSS Brute Force

This type of attack targets websites that contain programming vulnerabilities. The attacker injects malicious code into the website, which is executed when the user visits the compromised page. This code can steal the user's session ID, login credentials, or any other sensitive information.

1.3.4 SQL injection

SQL injection is a type of code injection that can cause damage to a database. These attacks are among the most common attacks on web applications.

SQL injection attacks occur when an attacker injects malicious SQL commands into input fields, allowing them to access, modify, or delete sensitive data from the database system.[9]

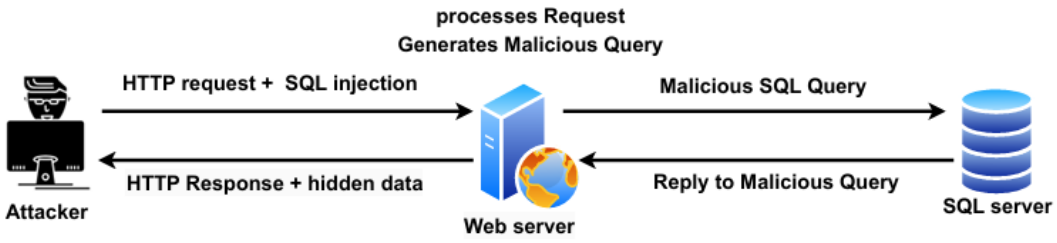


Figure 1.5: SQL injection

1.3.5 Infiltration

Infiltration is defined as unauthorized access to a digital environment with the aim of stealing information, disrupting services, or taking control of the targeted system. Attackers may rely on various methods, such as phishing, user manipulation, or exploiting vulnerabilities, to infiltrate systems undetected. Once the intrusion is successful, attackers can carry out other malicious activities, such as planting malware or leaking sensitive information. Preventing intrusion requires implementing effective security measures, including robust authentication mechanisms, training users on security awareness, and conducting regular audits of system vulnerabilities.

1.3.6 Botnet

A botnet is an interconnected network of internet-connected devices—including computers, smartphones, and IoT devices—that have been compromised by malicious software designed to allow remote control by a single malicious actor. These compromised systems, often referred to as "zombies," operate collectively under the control of the botnet operator to launch automated, large-scale cyberattacks that a single device could not carry out on its own.[10]

1.4 Network Security Measures

Securing a network can be achieved through several methods, the most common of which are covered below.

1.4.1 Firewalls

A firewall is a network security device that isolates the secure internal network from the untrusted external network, such as the Internet. It also manages incoming and outgoing network traffic based on predefined security rules. Firewalls are a critical component in protecting networks from unauthorized access, malicious behavior, and potential threats. They can take the form of physical hardware, applications, online software services, or be deployed on public or private cloud platforms.

Firewalls scrutinize network packets and implement security policies, effectively barring unauthorized users or potentially harmful data from infiltrating or exiting a network. Notably, firewalls serve as gatekeepers, scrutinizing each network packet and deciding whether to permit or block it based on pre-set rules. This helps to ensure that only traffic deemed safe and legitimate is allowed through the firewall.

In addition to these core functions, today's next-generation firewalls (NGFWs) have a range of features to bolster network security. These include deep packet inspection, application visibility and control, intrusion detection and prevention, malware defense, URL filtering, and more.

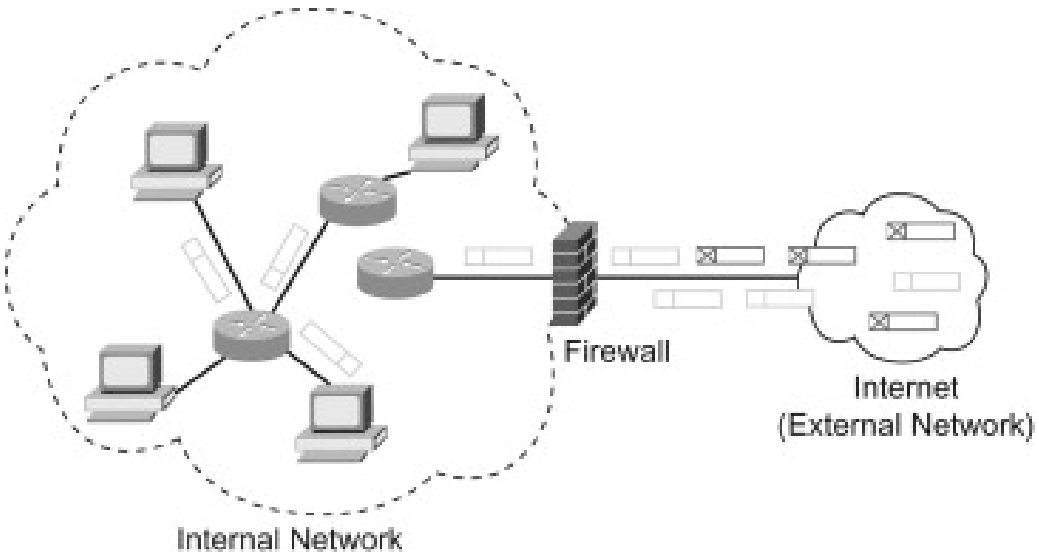


Figure 1.6: Firewall.

There are several types of firewalls commonly used in computer networks, including:

1.4.1.1 Packet filtering firewall

Firewalls analyze all network packets that pass through them and then filter them based on parameters such as the source and destination IP addresses, port numbers, and protocol types. Although these firewalls are simple and low-cost, they are unable to analyze packet content, which limits their effectiveness against sophisticated threats.

1.4.1.2 Stateful inspection firewall

A stateful inspection firewall is a type of traditional firewall that allows or blocks data flow based on the connection state, port, and connection type. It also monitors network traffic from the moment the connection is established. Filtering decisions are determined based on rules set by the administrator, as well as context, which refers to the use of information from previous sessions and packets associated with the connection itself.

1.4.1.3 Next-generation firewall (NGFW)

A Next-Generation Firewall (NGFW) is a network security system that offers additional features compared to traditional firewalls, which rely on stateful inspection. While a traditional firewall typically provides analysis of incoming and outgoing data traffic, a next-generation firewall includes advanced features such as application monitoring and

management, intrusion prevention (IPS), and link filtering based on geographic location and trustworthiness, in addition to threat intelligence. It also helps simplify management and reduce complexity through standardized rules across various stages of attacks.

1.4.2 Virtual Private Networks (VPNs)

A virtual private network (VPN) is an encrypted internet connection between a device and a network. This secure connection helps protect the transmission of sensitive information. It also prevents unauthorized users from eavesdropping on communications and enables remote work. VPN technology is widely used within organizations and companies.

1.5 Intrusion Detection System(IDS)

Intrusion detection refers to the process of monitoring and analyzing activities occurring on a computer or network to detect signs of intrusion, which are defined as attempts to compromise the confidentiality, integrity, or availability of data, or to bypass the security measures of a computer or network. Intrusions may result from hackers accessing systems via the Internet, from authorized users attempting to gain unauthorized additional privileges, or from users misusing the privileges granted to them.[11].

1.6 Types of IDS

There are many types of IDS, which can be summarized as follows[12]:

1.6.1 Network-based IDS

It monitors network traffic for particular network segments or devices and analyzes the network and application protocol activity to identify suspicious activity.It can identify many different types of events of interest. It is most commonly deployed at a boundary between networks, such as in proximity to border firewalls or routers, virtual private network (VPN) servers, remote access servers, and wireless networks.

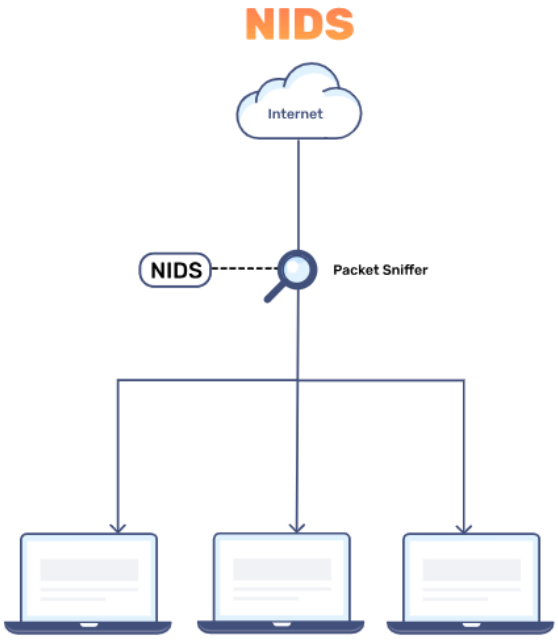


Figure 1.7: Network-based IDS.

1.6.2 Host-based IDS

It monitors the characteristics of a single host and the events occurring within that host for suspicious activity. Examples of the types of characteristics a host-based IDS might monitor are network traffic (only for that host), system logs, running processes, application activity, file access and modification, and system and application configuration changes. Host-based IDSs are most commonly deployed on critical hosts such as publicly accessible servers and servers containing sensitive information.

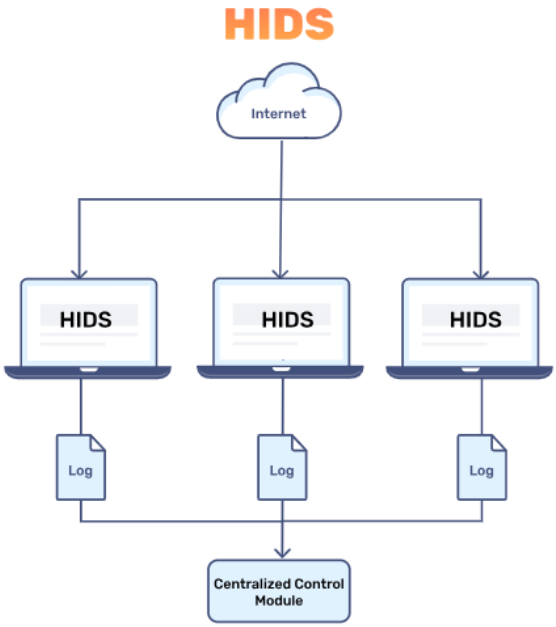


Figure 1.8: Host-based IDS

1.7 IDS Architecture

In general, an IDS contains three main components as depicted in Figure 1.9:

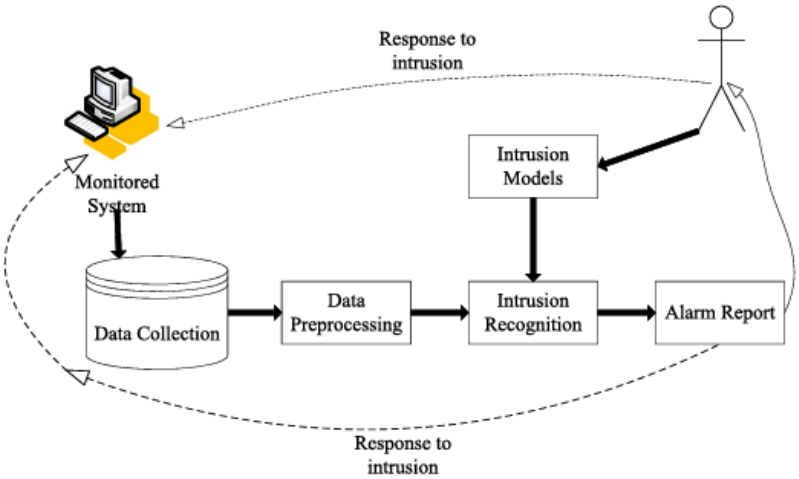


Figure 1.9: Framework of Intrusion Detection System

1.7.1 Data Collection

These sensors collect data from various sources, such as network traffic, system logs, and other sources that can be used to identify potential attacks. Intrusion detection systems typically rely on sensors to collect data from these sources, such as network-based intrusion detection systems, host-based intrusion detection systems, and application-based intrusion detection systems. Among the most prominent sensing technologies used in intrusion detection systems are Snort, Suricata, and Zeek.

1.7.2 Data pre-processor

Data preprocessing involves sorting, standardizing, and extracting features from the collected data, with the aim of creating activity data relevant to security analysis. After processing, the data may include information such as source and destination IP addresses, port numbers, protocol types, timestamps, and packet data. The goal of data preprocessing is to reduce the volume of data requiring analysis, enhance the quality of the remaining data, and make it more suitable for further analysis.

1.7.3 Intrusion recognition

Attack recognition is a process that identifies and classifies security risks detected by an intrusion detection system, using various techniques such as signature-based detection, anomaly-based detection, or machine learning-based detection, with the aim of identifying and classifying security threats.

1.8 conclusion

At the end of this chapter, the key concepts of network security and its three main pillars—confidentiality, integrity, and availability—were highlighted, as these constitute the fundamental elements upon which any effective security system is built. The most prominent security risks to networks were also explained, such as denial-of-service attacks (DDoS and DoS), brute-force attacks, SQL injection, intrusion, and botnets, which reflect the risks facing modern systems. In addition, the most important protection mechanisms used to secure networks, such as firewalls and virtual private networks (VPNs), were explained, as they significantly contribute to reducing the likelihood of attacks and pro-

tecting data in transit. Emphasis was also placed on intrusion detection systems as one of the most prominent modern protection technologies, through an explanation of their different types (network-based, host-based, and hybrid systems), as well as detection methods based on signature-based detection and anomaly-based detection. In conclusion, the architecture of the IDS system and its basic components—such as data collection, preprocessing, and intrusion detection—were explained.

Based on the above, we conclude that understanding these fundamentals represents a necessary step toward improving smart and effective security solutions capable of mitigating growing cyber risks.

Chapter 2

Background & Related Work

2.1 Introduction

Intrusion detection has become an important research field because of the rapid growth and increasing complexity of cyber threats, as well as the weaknesses of conventional security solutions. As explained in the previous chapter, modern network environments require smart and flexible intrusion detection systems capable of identifying malicious activities effectively.

This chapter presents the theoretical background and reviews the most relevant studies related to intrusion detection systems. First, traditional intrusion detection techniques are discussed, with a focus on how detection decisions are made in both signature-based and anomaly-based approaches, as well as their limitations. Next, machine learning methods applied to intrusion detection are reviewed, followed by deep learning approaches that have recently demonstrated promising performance in detecting complex network attacks. In addition, commonly used benchmark datasets for evaluating intrusion detection systems are presented. Finally, a comparative discussion of previous works is provided in order to identify existing research gaps and justify the motivation behind the proposed approach in this work.

2.2 Traditional Intrusion Detection Approaches

Traditional intrusion detection methods were the first techniques used to monitor network traffic and detect malicious activities. Before the emergence of machine learning approaches, these methods were widely adopted in intrusion detection systems.In general,traditional IDS techniques are divided into two main categories: anomaly-based detection, which identifies unusual behavior,and signature-based detection,which relies on known attack patterns to recognize threats[13].

2.2.1 Anomaly-Based Detection

In anomaly detection, we do not look for a known attack or a specific signature; rather, we look for suspicious activity in network traffic. The idea is that any abnormal behavior may be suspicious. Building this type of detection system begins by defining what constitutes a normal pattern for the target system or user, then determining the threshold at which activities will be considered unusual, along with the mechanism for making that decision. Consequently, this detection principle identifies behaviors that do not appear to result

from normal behavior, without relying on pre-known attack patterns.

2.2.1.1 Self-learning systems

Self-learning systems rely on analyzing samples to determine what constitutes normal behavior within the system. This is typically done by monitoring network traffic over an extended period of time and then building a model that represents the system's operational mechanisms and normal behavior.

A. Non-time Series

This concept refers to a set of indicators that analyze and represent the system's normal behavior using stochastic models, while ignoring behaviors associated with the time series of the data.

- **Rule Modelling:** In rule modelling, the system studies normal traffic and automatically derives a rule base describing its expected behaviour. At detection time, incoming traffic is matched against these rules in a weighted manner; a sufficiently poor match causes the system to trigger an alarm.
- **Descriptive statistics:** The descriptive statistics approach builds a profile of normal system behaviour by collecting simple statistical measurements from selected system parameters. At detection time, a distance vector is computed between the observed traffic and the stored profile. If the distance is sufficiently large, an alarm is raised.

B. Time series

This model is of a more complex nature, taking time series behaviour into account. Examples include such techniques such as a hidden Markov model (HMM), an artificial neural network (ANN), and other more or less exotic modelling techniques.

- **Artificial neural network(ANN):** The ANN approach operates as a black box model trained on normal traffic to learn its underlying patterns. During detection, new traffic is evaluated by the trained network, and its output is used to form the intrusion decision. In cases where the ANN output alone is insufficient, it is forwarded to a secondary expert system that makes the final classification.

2.2.1.2 Programmed

In programmed anomaly detection, a human operator explicitly defines the conditions under which the system should signal a security violation. Rather than learning from data autonomously, the system relies on human judgment to determine what constitutes abnormal behaviour worthy of an alarm.

A. Descriptive statistics

Programmed descriptive statistics systems establish a profile of normal system behaviour by collecting statistical measurements across a range of monitored parameters. These parameters typically include indicators such as the number of unsuccessful login attempts, the number of active network connections, and the number of commands that return error codes. The resulting profile serves as a reference baseline against which newly observed behaviour is compared in order to identify potential deviations that may indicate intrusive activity.

- **Simple statistics:** In all systems of this class, the collected statistics are not used directly to raise an alarm. Instead, they are forwarded to higher level components within the detection architecture, which process and analyse them at a more abstract level in order to formulate the final intrusion detection decision.
- **Simple rule-based:** In simple rule-based systems, the human operator provides the system with a set of compound logical rules to be applied to the previously collected statistical measurements. Although these rules are relatively straightforward in nature, they are capable of combining multiple conditions simultaneously, allowing the system to evaluate complex behavioural patterns. When an observed set of statistics satisfies a defined rule condition, the system triggers an alarm.
- **Threshold:** Threshold-based detection represents the simplest form of programmed descriptive statistics detector. Once the system has collected the necessary statistical measurements, the human operator defines a set of predefined thresholds, which may take the form of simple numerical ranges, to govern the alarm triggering process. The decision mechanism is direct: if a monitored parameter exceeds its predefined threshold, the system immediately raises an alarm. A typical example of such a rule is: raise an alarm if the number of unsuccessful login attempts exceeds three.

B.Default deny

In the default deny approach, the operator explicitly specifies all circumstances under

which the system is considered to be operating normally. Any deviation from these defined conditions is automatically flagged as intrusive. This mirrors a default deny security policy, analogous to legal systems that define permitted behaviour and label everything else as illegal.

- **State series modelling :** In state series modelling, the policy defining security-benign operation is encoded as a sequence of states. Unlike explicit state machine implementations, the transitions between these states are implicit within the model rather than formally declared. Once the system matches a given state, the detection engine waits for the next expected transition to occur. If the monitored action corresponds to a permitted transition, the system continues its observation without intervention. However, if the observed action leads to a state that is not explicitly defined in the model, the system immediately raises an alarm. The types of actions capable of triggering state transitions are typically security-relevant operations, such as file read and write accesses and the opening of privileged communication ports. The rule matching engine employed in this approach is simpler and less powerful than a full expert system, as it does not support unification. It does, however, support fuzzy matching, whereby a broadly defined condition such as write access to any file within a specified directory can trigger a transition. This flexibility is essential for making the specification of permitted behaviour practically feasible, as an overly rigid matching mechanism would render the complete definition of normal operation unrealistic.

2.2.2 Signature-Based detection

Signature-based detection takes a completely different direction compared to anomaly detection. The system here has no interest in what normal traffic looks like. Its only job is to find traces of attacks that have been seen and studied before.

The idea behind it is not complicated. Security experts analyse real attacks and describe what they look like in terms of the patterns and traces they leave on the system. These descriptions become the signatures. When new traffic comes in, the system checks it against these signatures. A match means an alarm. No match means nothing happens and monitoring continues.

That said, this method has one big weakness. Everything depends on how well the signature is written. A signature that is too broad will keep triggering false alarms. One that is too narrow will miss variations of the same attack. There is no room for mistakes

here because in a real network environment, both missing an attack and flooding analysts with false alerts can have serious consequences.

A.Programmed

In this approach, the system is given a clear and explicit decision rule from the start. The person who builds the system has already done the hard work of filtering out any noise or interference, so what reaches the detection engine is clean and ready to be evaluated.

The rule itself is not complicated. It simply describes what you would expect to see if a specific attack was taking place. The idea is to be very precise about which traces are unique to that attack and would not normally appear in regular traffic.

This way of thinking is very similar to how the law works. Instead of defining everything that is allowed, you list what is illegal, and everything else is automatically considered fine. In security terms, this is called a default permit policy. If the observed behaviour does not match any of the listed attack patterns, the system does nothing and lets it through.

- **State-modelling :** An attack rarely happens all at once. It usually goes through several stages, and state modelling is built around this fact. For the system to flag something as an intrusion, every required stage must actually show up in the monitored traffic. Two variations exist within this approach.

  The first one, state transition, assumes the attack follows a fixed sequence. Each stage must come after the previous one, no skipping and no reordering. Once all stages are completed in that exact order, the system triggers an alarm.

  The second one, petri-net, is less strict about ordering. The attack still needs all its stages to be present, but they can appear in any sequence. This works better for attacks that behave differently each time while still going through the same general steps.

- **Expert-system :** Expert systems go beyond simple pattern matching. They use a rule base describing attack behaviour and reason through the available evidence to decide whether something malicious is happening. Forward chaining is the most common technique used here. The system treats incoming audit events as facts and keeps firing rules against them as new events arrive. This suits intrusion detection well because the stream of events never really stops.

  These systems are also quite capable compared to other methods. Unification, for example, lets them work with abstract and complex conditions that basic detectors

cannot handle. This gives analysts a lot of room to express sophisticated attack definitions.

The downside is speed. All that reasoning takes time, and in environments where traffic is heavy, this can become a real problem. Simpler methods will always be faster, so there is always a tradeoff to think about when choosing this approach.

- **String matching:** String matching works in a very simple way. The system scans through transmitted text or system logs and checks whether any known malicious substrings appear. If something matches, an alarm goes off. Nothing more to it. One thing worth mentioning is that the matching is usually case sensitive. A single difference in capitalisation is enough for the system to completely miss the threat. This points to the main weakness of this method, which is its rigidity. Attackers do not need to do much to get around it. Changing one character in the attack string is often enough. Still, not every situation needs a sophisticated solution. String matching is quick to set up, easy to follow, and supported by many efficient search algorithms that handle large volumes of audit data without much trouble. In environments where simplicity and speed are the priority, it can still do a reasonable job.

- **Simple rule-based :** Simple rule-based systems follow a similar logic to expert systems, but they are less complex.This simplicity helps them run faster.In environments where response time matters, this can make a real difference.

2.2.3 Hybrid IDS

A hybrid intrusion detection system combines both anomaly-based and signature-based detection methods to address the limitations of each approach. A hybrid system leverages signature-based detection for known threats and anomaly-based detection for novel attacks. This enhances the overall effectiveness of intrusion detection[14]

2.3 Machine Learning Approaches

Several studies have applied classical machine learning algorithms to the intrusion detection problem. This section reviews the most relevant works, focusing on the models used, the datasets evaluated, and the results obtained.

Ajagbe et al. (2024) ran an experiment comparing five machine learning classifiers

on the UNSW-NB15 dataset, which was put together at the Austrian Centre for Cyber Security. The data has around 2.5 million records spread across 49 features. One thing that stood out about this dataset was how unbalanced it was. Normal traffic took up about 93% of the records, with attacks sitting at just 7%. That kind of gap makes training tricky since the models barely see enough attack examples to learn from. Before any training happened, the authors went through a cleaning step to drop features that would basically hand the model the answer. They then normalized the data using min-max scaling and used oversampling to bring the attack class up a bit. The data was split 70/15/15 for training, validation, and testing. The classifiers they picked were Linear SVC, Logistic Regression, Decision Tree, Random Forest, and XGBoost. They scored each one on accuracy, precision, recall, F1, and AUC-ROC. Table2.1 has the full breakdown.

Model	Accuracy	Precision	Recall	F1-Score	AUC-ROC
Linear SVC	68.00%	68.44%	68.00%	68.08%	0.91
Logistic Regression	69.41%	69.84%	69.81%	69.49%	0.70
Decision Tree	83.01%	83.28%	83.00%	83.05%	0.83
XGBoost	89.48%	89.68%	89.48%	89.51%	0.90
Random Forest	**90.39%**	**90.58%**	**90.39%**	**90.41%**	**0.91**

Table 2.1: Performance of ML models on UNSW-NB15 – Ajagbe et al. (2024) [15]

Obagbuwa et al. (2026) worked with the CICIDS2017 dataset, which covers five days of network traffic recorded in July 2017. The data includes several attack categories such as DoS/DDoS, brute force, port scan, web attacks, and botnet traffic. The five daily files were merged into one before any processing took place. From there, 308,381 duplicate records were dropped. SMOTE was then applied to fix the class imbalance, and the features were normalized and encoded to get the data ready for training. Out of the four models the authors tested, three fall under classical machine learning: Naive Bayes, Random Forest, and XGBoost. The evaluation relied on 10-fold cross-validation backed by bootstrapped 95% confidence intervals. Accuracy, precision, recall, and F1-score were used to measure how well each model performed. Table 2.2 has the numbers.

Model	Accuracy	Precision	Recall	F1-Score
Naïve Bayes	0.93 ± 0.02	0.94 ± 0.02	0.93 ± 0.02	0.93 ± 0.02
Random Forest	0.96 ± 0.01	0.96 ± 0.01	0.96 ± 0.01	0.96 ± 0.01
XGBoost	$\mathbf{0.997 \pm 0.004}$	$\mathbf{0.997 \pm 0.003}$	$\mathbf{0.997 \pm 0.003}$	$\mathbf{0.997 \pm 0.003}$

Table 2.2: Performance of classical ML models on CICIDS2017 -Obagbuwa et al.(2026)[16].

2.4 Deep Learning Approaches

As the limitations of classical machine learning became more apparent, researchers began shifting towards deep learning models, which are capable of automatically learning complex patterns from raw data without relying heavily on manual feature engineering. This section reviews the most relevant deep learning-based studies in the context of network intrusion detection, focusing on the architectures proposed, the datasets used, and the classification performance achieved.

In their **2022** study, **Qazi et al**. approached the CICIDS2017 dataset by merging all eight daily files, producing a combined dataset of over 2.8 million instances. After removing non-informative identifiers and filtering out corrupted records, 79 features remained. To deal with severe class imbalance, random under-sampling brought the working dataset down to 135,000 instances across five categories. Their deep learning model followed a 1D-CNN structure, leveraging five convolutional layers to extract sequential patterns from the network flow data, supported by dropout regularisation and max-pooling to improve generalisation. The model was optimised using Adam over 50 epochs and evaluated on a held-out 25% test split. Table 2.3 summarises the classification performance.

Phase	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Training	99.32	99.2	99.5	99.34
Testing	98.96	98.7	99.2	98.94

Table 2.3: Performance of Deep Learning model on CICIDS2017 – Qazi et al. (2022)[17].

In their**2022** study **Farhan and Jasim** carried out their study using the CSE-CIC-IDS2018 dataset, which was generated from real network traffic collected through the Amazon Web Services (AWS) environment. The dataset includes more than 1.25 million records and covers ten different traffic categories, including normal traffic as well as several attack types such as Bot, FTP-BruteForce, SSH-Bruteforce, DDoS-HOIC, DDoS-LOIC-UDP, DoS-GoldenEye, DoS-SlowHTTPTest, DoS-Hulk, and Infiltration.

Before training the model, the authors performed several preprocessing operations. Features that did not contribute to the learning process, including timestamps and IP-related identifiers, were removed, reducing the number of features from 80 to 78. The remaining data was then normalized within the range [-1,1] to improve the training process. After preprocessing, the dataset was divided into 70% for training and 30% for testing.

To detect malicious traffic, the researchers adopted a deep learning approach based on a three-layer Long Short-Term Memory (LSTM) network. The model was trained using the Adam optimizer for 30 epochs with a batch size of 200. According to the reported results, the proposed method achieved an accuracy of 99%, showing better performance than several other techniques evaluated on the same dataset. The detailed results are presented in Table 3.25.

Research	**DL and ML**	**Data set**	**Accuracy**
Lin *et al.* [18]	LSTM + AM	CSE-CIC-IDS 2018	96.2%
Seth *et al.* [19]	Light GBM + HBGB	CSE-CIC-IDS 2018	97.5%
Rios *et al.* [20]	CFBLS and BLS	CSE-CIC-IDS 2018	97.46%
The proposed LSTM [21]	LSTM	CSE-CIC-IDS 2018	99%

Table 2.4: The comparison among LSTM model and another method

2.5 Conclusion

This chapter covered the main concepts and prior work relevant to network intrusion detection.We began by examining the two core detection approaches,anomaly-based and signature-based,each with its own strengths and limitations.Anomaly detection is flexible enough to catch unknown threats, but tends to produce more false alarms. Signature-based detection is reliable against known attacks,but cannot handle anything it has not seen before.

We then reviewed a number of studies that applied classical machine learning to this problem. Models like Random Forest and XGBoost showed solid results, but their performance depends heavily on feature engineering and they tend to struggle with the complexity of modern network traffic.

More recent work has moved towards deep learning,and the results have been encouraging.CNN and LSTM-based models have demonstrated strong detection capabilities across several benchmark datasets.That said, most of these studies were evaluated on a single dataset,which makes it difficult to draw firm conclusions about how well they would hold up in different environments.

This gap is part of what motivated the work presented in this thesis,and the following chapters describe how we addressed it.

Chapter 3

Building and Evaluating Machine Learning and Deep Learning Models for IDS

3.1 Introduction

In this work, dataset selection was a crucial step, as the performance of any deep learning model depends heavily on the quality of the data used during training. Therefore, three different datasets were used: CSE-CIC-IDS2018, CIC-IDS2017, and the Network Intrusion Dataset. This provided diverse data encompassing various types of cyberattacks, in addition to normal network traffic.

These datasets contained a large volume of data representing diverse and realistic attack scenarios, such as DoS, DDoS, and brute-force attacks. This enabled the model to learn more effectively and recognize different attack patterns. Furthermore, using multiple datasets allowed the model to handle more diverse datasets, rather than relying on a single dataset.

Initially, the model was implemented in a preliminary manner to test its performance and gain a general understanding of the results. The model was then evaluated using several metrics such as Accuracy, Precision, Recall and F1-score. Based on the results obtained, some improvements were made to the work method and training with the aim of improving the model's performance and achieving better results in the process of detecting and classifying cyberattacks.

3.2 Execution environment

Deep Learning relies heavily on hardware to execute tasks effectively. There are a few fundamental minimum hardware requirements that must be fulfilled before Deep Learning can begin [22]. The basic minimum hardware requirements to start are:

1. Graphics Processing Unit (GPU): Intel HD Graphics 620 (integrated, not NVIDIA).
2. RAM:16GB and above.
3. HDD/SSD:256GB SSD only (no HDD).
4. CPU:Intel i5-7200U (7th Gen)

Even with all the basic hardware requirements being met, dealing with large datasets can still be a time-consuming task. However, numerous cloud computing resources are available that offer more RAM, CPUs, and GPUs either for free or for a limited period

of time. These resources make data processing more efficient and significantly reduce the occurrence of errors. Therefore, Kaggle Kernels were used as both a hardware and software alternative for this work.

Kaggle [23] is an online community platform that provides a range of tools and resources for data scientists, Machine Learning practitioners, and researchers. It of fers a variety of services, including cloud-based Jupyter notebooks, GPU and TPU resources, data visualization tools, and more, all aimed at helping users work with data more efficiently. In terms of software, Kaggle offers a broad range of tools and packages commonly used in data science and Deep Learning. This includes Jupyter Notebook along with popular programming language ***Python***[24], alongside its widely-used libraries such as NumPy, Pandas, Scikit-learn, Tensor-Flow, and Keras. Jupyter Notebook is a web-based interactive computing environment provided by Kaggle that allows users to create and share codes, visualizations, and texts in a single document. It is a popular tool among data scientists and Deep Learning practitioners, as it supports multiple programming languages such as Python. Python is a widely used, high-level, interpreted programming language that is open source and offers an excellent approach to object-oriented programming. It is a popular deep learning language used by data scientists for a variety of Deep Learning projects and applications. With its well-defined syntax and dynamic typing, Python is a powerful tool for manipulating complex data structures and algorithms, making it an ideal language for developing Deep Learning models.

Library	Description	Version
Pandas [25]	Pandas is a Python library that provides data manipulation and analysis tools. It is widely used for data cleaning, data transformation, data exploration and data visualization.	1.3.5
TensorFlow [26]	TensorFlow is an open source library focused on deep neural network training, providing options to deploy deep learning models locally to a database, or to the cloud. It is an indispensable tool for the modern data scientist due to its advanced features and support for building and training deep learning models using the latest technologies.	2.11.0
NumPy [27]	NumPy is a Python library that provides support for large multidimensional arrays and matrices. It is commonly used in scientific computing and numerical analysis.	1.21.6
Scikit-learn [28]	Scikit-learn is a Python library that provides machine learning algorithms for classification, regression, clustering, and dimensionality reduction. It is commonly used in data analysis and predictive modeling.	1.0.2
Matplotlib [29]	Matplotlib is a Python library that provides data visualization tools. It is commonly used to create charts, histograms, scatterplots, and other visualizations to aid in understanding and communicating data.	3.5.3

Table 3.1: List of python libraries

3.3 Evaluating the proposed models on merging CIC-IDS2017 and CSE-CIC-IDS2018 and Network Intrusion Dataset datasets

3.3.1 Data Sets

3.3.1.1 Description of Intrusion detection dataset CIC-IDS2017

The CICIDS2017 dataset is one of the most up-to-date and important databases used in the field of intrusion detection systems. It was developed by the Canadian Institute of Cyber Security to provide comprehensive, real-world data that helps evaluate network attack detection models. This dataset comprises two types of traffic: normal and offensive. Both were generated within a simulated environment closely resembling real networks and include various types of attacks such as DDoS, DoS attacks, brute-force attacks, botnet attacks, web attacks, infiltration, and Heartbleed attacks. The data was collected over five consecutive days, from July 3 to 7, 2017, with Monday dedicated to normal traffic and the other days to attack traffic. The dataset also contains over 80 properties extracted using the CICFlowMeter tool, along with highly categorized PCAP and CSV files, making it suitable for training and testing AI and deep learning models in the field of intrusion detection. CICIDS2017 is distinguished by the diversity of its protocols, the realism of its data, the variety of attack types it covers, and its comprehensive network traffic logging, making it an important reference in many recent cybersecurity studies[30].

File Name	Day Activity	Attacks found
MondayWorkingHours.pcap_ISCX.csv	Monday	Benign
WorkingHours.pcap_ISCX.csv	Tuesday	Benign, BruteForce
WednesdayworkingHours.pcap_ISCX.csv	Wednesday	Benign, DoS
Thursday-WorkingHoursMorning-WebAttacks.pcap_ISCX.csv	Thursday	Benign
Thursday-WorkingHoursAfternoon-Infilteration.pcap_ISCX.csv	Thursday	Benign, Infiltration
Friday-WorkingHoursMorning.pcap_ISCX.csv	Friday	Benign, Botnet
Friday-WorkingHoursAfternoonPortScan.pcap_ISCX.csv	Friday	Benign, PortScan
Friday-WorkingHoursAfternoonDDos.pcap_ISCX.csv	Friday	Benign, DDoS

Table 3.2: CICIDS2017 Files after Label Normalization

3.3.1.2 Descriptions of CSE-CIC-IDS2018 dataset

The CSE-CIC-IDS2018 dataset on AWS is a network attack detection dataset containing large-scale real network environments with millions of network packets. The dataset contains various types of attacks and normal activities, and provides a wide range of features such as timestamps, packet and flow level details, and network behavior patterns. The CSE-CIC-IDS2018 dataset is used to develop and evaluate new intrusion detection algorithms, test and validate cybersecurity tools, and provide valuable insights into the latest cyberattack trends and patterns. Due to its large size and quality, this collaborative project between the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC) was the perfect choice for our work[32].

Files	Day Activity	Attacks found
02-14-2018.parquet	Wednesday	Benign Brute Force Other
02-15-2018.parquet	Tuesday	Benign DoS Other
02-16-2018.parquet	Friday	DoS Hulk Benign Other
02-20-2018.parquet	Tuesday	Benign DDoS attacks
02-21-2018.parquet	Wednesday	DDoS Benign Other
02-22-2018.parquet	Thursday	Benign Brute Force Other
02-23-2018.parquet	Friday	Benign Brute Force Other
02-28-2018.parquet	Wednesday	Benign Infiltration Other
03-01-2018.parquet	Thursday	Benign Infiltration Other
03-02-2018.parquet	Friday	Benign Bot

Table 3.3: Description of files containing the CIC-IDS2018 dataset

3.3.2 Overview of the work pipeline

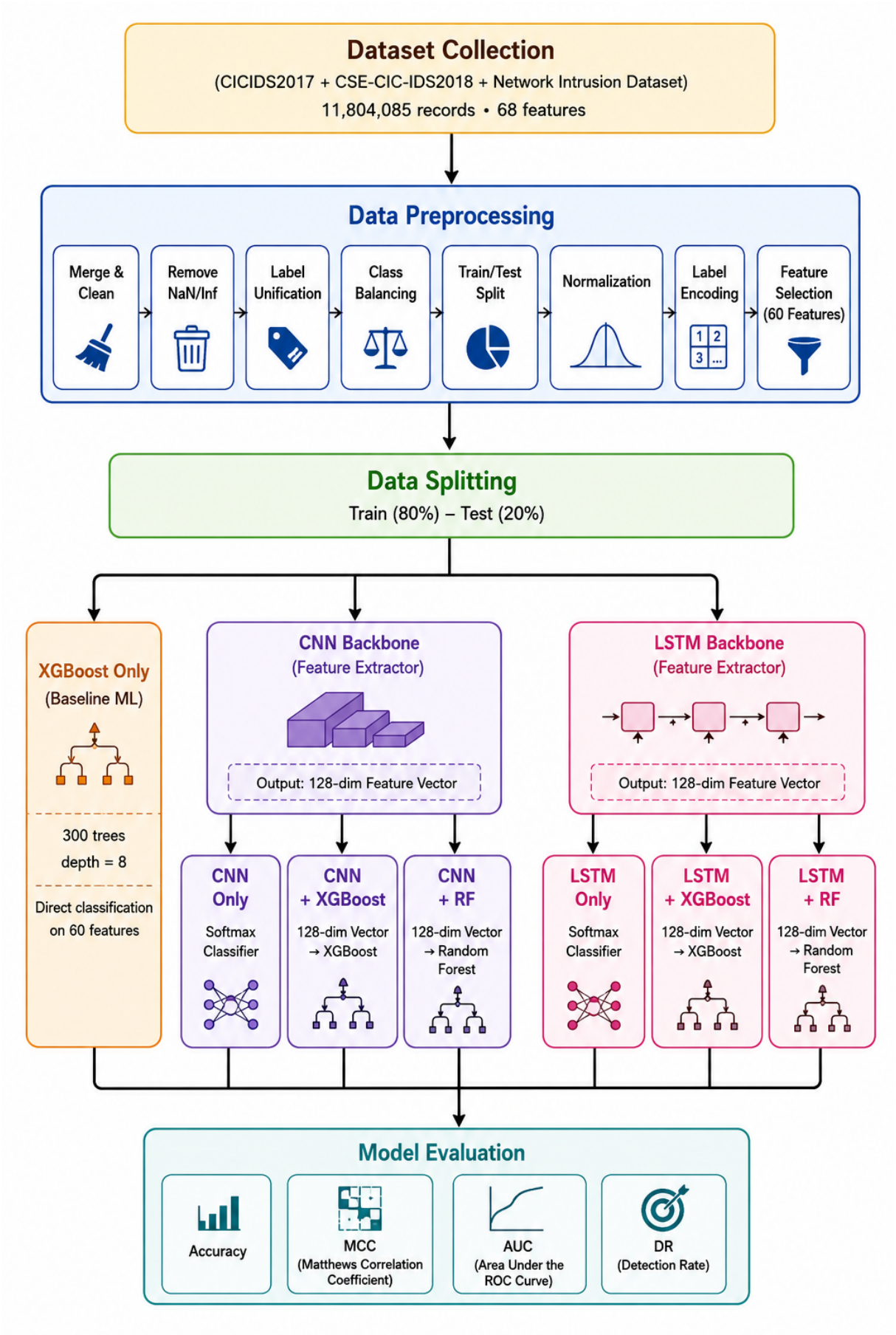


Figure 3.1: Component architecture of the proposed work

3.3.3 Data Preprocessing

When utilizing raw network traffic data, you will undoubtedly face numerous practical obstacles to addressing before you get any value out of your learning experiments. The collected network traffic data will come in from many different sources and shall therefore be inherently heterogeneous, composed of missing values, labelled inconsistently, with redundant features, and with class imbalance between normal and attack traffic. If these obstacles remain unresolved, they would adversely and substantially degrade the performance of any machine learning or deep learning algorithm trained on that data.

To ensure fair and rigorous evaluation across all proposed models, a single preprocessing pipeline was created and consistently applied to the data prior to processing the data according to whatever model was being used. The preprocessing pipeline consists of the following separate but well-defined processes whereby the raw device data undergoes a series of transformations — merge, label unify, balance, clean, split, encode, select features, normalise — each of which are described in detail in the subsequent subsections of this document.

One of the key design principles throughout the preprocessing pipeline is adhering to the principle of avoiding data leakage; any transformation or modelling exercise that includes fitting any statistical model (e.g., scaling or selecting features) is done only on the training portion of the data set and later applied to the unseen test data so as to accurately represent the true level of generalisation exhibited by the model based upon unseen data.

3.3.3.1 Data Collection and Merging

The dataset used in this study was assembled from three publicly available benchmark datasets that are widely used in the network intrusion detection research community: CICIDS2017, CSE-CIC-IDS2018, and the Network Intrusion Dataset. These datasets were selected because they collectively cover a broad spectrum of attack types and were generated under controlled but realistic network conditions, making them well-suited for evaluating intrusion detection models. Together, the three sources contributed 26 separate files stored in a mix of Parquet and CSV formats, which were loaded and consolidated into a single unified dataset.

One of the first practical challenges encountered during this stage was the inconsistency in feature naming across the different sources. Some column names contained

leading or trailing whitespace characters, which would have caused mismatches during merging. This was resolved by stripping all column names before any further processing. A more fundamental challenge was that the three datasets did not share identical feature sets; each source was generated using a slightly different version of the CICFlowMeter tool, resulting in minor differences in the available columns. To handle this, only the features that appeared in all three sources were retained, ensuring a consistent and coherent feature space across the merged dataset. After merging, the combined dataset contained **11,804,085 records** with **68 features**, providing a rich and diverse foundation for model training and evaluation.

3.3.3.2 Label Unification and Class Filtering

Each of the three datasets was originally annotated using its own naming convention for attack categories. For instance, what one dataset labeled as *DoS Hulk* or *DoS GoldenEye* was functionally the same type of attack as what another labeled *DoS attacks-Hulk* or *DoS attacks-GoldenEye*. Similarly, brute-force attacks appeared under names such as *FTP-Patator*, *SSH-Patator*, *FTP-BruteForce*, and *SSH-Bruteforce* across different files. If left unaddressed, this fragmentation would have caused the model to treat semantically identical attacks as distinct classes, introducing noise and making the classification problem unnecessarily difficult.

To resolve this, a comprehensive label mapping was applied to consolidate all variant names into a unified set of seven traffic classes. The mapping is summarized in Table 3.4.

Original Labels			Unified Class
CIC-IDS2017	**CIC-IDS2017 (CSV)**	**CSE-CIC-IDS2018**	
Benign	BENIGN	Benign	Benign
Bot	Bot	Bot	Botnet
DDoS	DDoS	DDoS attacks-LOIC-HTTP, DDOS attack-LOIC-UDP, DDOS attack-HOIC	DDoS
DoS slowloris, DoS Slowhttptest, DoS Hulk, DoS GoldenEye	DoS slowloris, DoS Slowhttptest, DoS Hulk, DoS GoldenEye	DoS attacks-GoldenEye, DoS attacks-Slowloris, DoS attacks-SlowHTTPTest, DoS attacks-Hulk	DoS
FTP-Patator, SSH-Patator	FTP-Patator, SSH-Patator	FTP-BruteForce, SSH-Bruteforce	BruteForce
PortScan	PortScan	—	PortScan
Infiltration	Infiltration	Infilteration	Infiltration
Heartbleed, Web Attack ∼ Brute Force, Web Attack ∼ XSS, Web Attack ∼ Sql Injection	Heartbleed, Web Attack ∼ Brute Force, Web Attack ∼ XSS, Web Attack ∼ Sql Injection	Brute Force -Web, Brute Force -XSS, SQL Injection	*Removed*

Table 3.4: Label Unification Mapping Across Datasets

Two classes were excluded from the study entirely. The *Heartbleed* class contained an extremely small number of samples — too few to support reliable model training or evaluation. The *WebAttack* class, while present in some files, was represented inconsistently across sources and could not be cleanly unified. Removing these classes allowed the study to focus on seven well-represented and practically relevant traffic categories.

3.3.3.3 Class Balancing via Downsampling

A persistent challenge in intrusion detection datasets is the severe imbalance between the volume of normal traffic and attack traffic. In the merged dataset, the Benign class accounted for the vast majority of records, vastly outnumbering several attack classes. Training a model directly on such skewed data tends to produce classifiers that are biased toward the majority class, achieving high overall accuracy while failing to detect minority attacks reliably.

To address this, a controlled random downsampling strategy was applied. Each class was capped at a predefined maximum number of samples, chosen to reduce memory consumption while maintaining a sufficiently representative distribution. For classes that were already below their respective cap, all available samples were retained in full to preserve every attack instance. The resulting distribution after downsampling is shown in Table 3.5.

Class	**Number of Samples**
Benign	300,000
DDoS	300,000
DoS	300,000
PortScan	150,000
Botnet	120,000
BruteForce	117,086 (all available)
Infiltration	118,555 (all available)
Total	**1,405,023**

Table 3.5: Class Distribution After Downsampling

This approach brought the total dataset size to approximately 1.4 million records — a scale that remained computationally manageable while still providing enough data for robust model training. The random sampling was performed with a fixed seed to ensure reproducibility across all experiments.

3.3.3.4 Data Cleaning

After balancing the class distribution, the next step was to ensure the quality and integrity of the feature data itself. Several categories of problematic values were systematically identified and removed.

First, a set of columns that carry no predictive value for intrusion detection were dropped from the dataset. Specifically, the columns `Flow ID`, `Src IP`, `Dst IP`, `Source IP`, `Destination IP`, and `Timestamp` were removed. These fields describe the identity and timing of individual connections rather than their behavioral characteristics, and including them would risk introducing irrelevant noise or, worse, overfitting to specific IP addresses or time windows present only in the training data.

Second, all remaining feature columns were explicitly cast to numeric format. Any values that could not be converted — typically arising from parsing artifacts or data entry errors — were coerced to `NaN`. Third, records containing infinite values ($\pm\infty$) or missing values (`NaN`) were removed entirely, as such entries would cause numerical instability during training. Finally, any feature column that contained only a single unique value across the entire dataset was dropped, since constant features contribute no discriminative information and serve only to add unnecessary dimensions to the input space. After completing all cleaning steps, the dataset retained **1,405,023 records** and **60 clean, informative features**.

3.3.3.5 Train/Test Split

One of the most important principles in machine learning model evaluation is ensuring that the test set remains completely unseen during every stage of model development, including preprocessing. Violating this principle — a problem commonly referred to as *data leakage* — inflates performance metrics during evaluation and leads to models that appear to generalize well but fail in practice. To strictly prevent this, the dataset was divided into training and test sets **before** applying any further transformations.

A stratified random split was used, allocating 80% of the data to the training set and 20% to the test set. Stratified splitting ensures that the proportion of each class is preserved in both subsets, which is particularly important given the varying class sizes in this study. The resulting split sizes are summarized in Table 3.6.

Subset	**Size**
Training Set	1,124,018 samples (80%)
Test Set	281,005 samples (20%)

Table 3.6: Train/Test Split

All subsequent steps — label encoding, feature selection, normalization, and validation splitting — were performed exclusively on the training set and then applied to the test

set using the parameters learned from training data only.

3.3.3.6 Label Encoding

Machine learning and deep learning algorithms operate on numerical representations of data. Since the class labels in the dataset are categorical strings such as *Benign* or *DDoS*, they must be converted into integer values before they can be used as model targets. This was accomplished using standard Label Encoding, which assigns a unique integer to each class in alphabetical order. The resulting mapping is shown in Table 3.7.

Class Label	Encoded Value
Benign	0
Botnet	1
BruteForce	2
DDoS	3
DoS	4
Infiltration	5
PortScan	6

Table 3.7: Label Encoding Mapping

As with all other fitted transformations, the encoder was fitted exclusively on the training labels and then applied to the test labels, ensuring that the encoding scheme was determined solely by the training data.

3.3.3.7 Feature Selection

Although the cleaned dataset retained 68 features, not all of them contribute equally to the task of distinguishing between different traffic classes. Including redundant or weakly informative features can slow down training, increase the risk of overfitting, and reduce the model's ability to generalize to new data. To identify and retain only the most discriminative features, `SelectKBest` with *Mutual Information* scoring was applied, selecting the top $k = 60$ features from the available set.

Mutual Information is a non-parametric measure that quantifies the statistical dependency between a feature and the target variable. Unlike correlation-based measures, it is capable of capturing nonlinear relationships, making it particularly well-suited for network traffic data where the relationship between flow statistics and attack type can

be highly complex. The feature selector was fitted on the training set only, ensuring that no information from the test set influenced feature selection decisions.

3.3.3.8 Normalization

Network traffic features vary enormously in their numerical scale. Packet lengths may reach values in the thousands, while flag counts remain in single digits, and flow duration values span several orders of magnitude. Without normalization, features with large magnitudes tend to dominate the learning process in gradient-based models, causing the optimizer to focus disproportionately on certain dimensions at the expense of others. This is especially problematic for deep learning architectures such as CNN and LSTM, where gradient-based updates directly depend on the scale of the input.

To eliminate this scale bias, all selected features were standardized using `StandardScaler`, which transforms each feature to have zero mean and unit variance according to the formula:

$$X_{\text{normalized}} = \frac{X - \mu_{\text{train}}}{\sigma_{\text{train}}} \tag{3.1}$$

where μ_{train} and σ_{train} are the mean and standard deviation computed from the training set, respectively. The scaler was fitted on the training data and then applied to both the training and test sets, ensuring that the test set was scaled using only statistics derived from the training distribution.

3.3.3.9 Validation Split and Class Weighting

After normalization, the training set was further partitioned into a training subset and a validation subset using a stratified 80/20 split. The validation set was not used for final evaluation; instead, it served as an independent signal during model training for two specific purposes: monitoring the validation loss to trigger early stopping when the model began to overfit, and guiding the learning rate scheduler to reduce the learning rate when improvement stalled. This approach ensured that the final model configuration was selected based on performance on held-out data rather than simply on training loss.

To further mitigate the effect of residual class imbalance after downsampling, class weights were computed using the balanced weighting strategy. This method assigns each class a weight inversely proportional to its frequency in the training set, causing the

loss function to penalize misclassifications of underrepresented classes more heavily. This was particularly important for classes such as *Infiltration*, which — despite downsampling efforts — remained challenging to distinguish from normal Benign traffic due to the inherently subtle nature of infiltration attack patterns.

3.3.4 Description and Architecture of the Proposed Models

Building an effective intrusion detection model requires more than selecting a powerful algorithm — it demands careful architectural decisions, appropriate training strategies, and a well-designed evaluation protocol. In this section, we describe in detail how each of the six proposed models was constructed, trained, and prepared for evaluation. All models were implemented using the TensorFlow/Keras framework for the deep learning components, scikit-learn for preprocessing utilities, and XGBoost and scikit-learn's ensemble module for the classical classifier components. The implementation was carried out on the Kaggle cloud computing platform, utilizing CPU-based execution throughout all experiments.

3.3.4.1 CNN-Based Classification Model

The CNN model is built as a complete end-to-end deep classifier. It takes 60 x 1 input (60 filtered features with one channel) and passes it through two consecutive Conv1D layers. The first layer uses 32 kernel-size filters, followed by BatchNormalization to stabilize distributions and MaxPooling size 2 to reduce length. The second layer uses the same architecture but with 64 filters to capture deeper patterns. The data is then flattened and passed through a Dense layer with 128 units and ReLU enabled, followed by a 0.5 Dropout to prevent overtraining, and finally a Softmax layer with 7 units for classification into seven categories.

Training: Optimizer Adam is used with a learning rate of 3×10^{-4} and a sparse categorical cross-entropy loss, with 15 epochs and a batch size of 2048. EarlyStopping is used to stop training after 3 epochs without improvement, and ReduceLROnPlateau is used to reduce the learning rate when improvement stops.d representation to a probability distribution over the seven traffic classes.

The complete architecture is summarized in Table 3.8.

Layer	Configuration	Output Shape
Input	Shape (60, 1)	(60, 1)
Conv1D	32 filters, kernel=3, ReLU, same	(60, 32)
BatchNorm	—	(60, 32)
MaxPooling1D	pool size=2	(30, 32)
Conv1D	64 filters, kernel=3, ReLU, same	(30, 64)
BatchNorm	—	(30, 64)
MaxPooling1D	pool size=2	(15, 64)
Flatten	—	(960,)
Dense	128 units, ReLU	(128,)
Dropout	rate=0.5	(128,)
Dense (output)	7 units, Softmax	(7,)

Table 3.8: CNN Only — Layer Configuration

3.3.4.2 CNN + XGBoost (CNN–XGB)

The CNN–XGB model is a two-stage hybrid architecture. Stage 1: The same CNN as the previous model, but without the final Softmax layer—instead, we extract 128 features from the final Dense layer. Stage 2: We feed these features to XGBoost with a 300-tree structure, a maximum depth of 8, a learning rate of 0.05, and a histogram to accelerate computation. The idea is that the CNN learns to extract hidden clues, and XGBoost learns the best decision rules based on these clues.

Training and Inference: First, we train the CNN using the CNN Only method until convergence. Then, we freeze the network weight (which doesn't change during XGBoost) and extract features from all the training data. We then train XGBoost on these features separately. When inferring new data, it first passes through the CNN to produce 128 features, and then XGBoost classifies them.

3.3.4.3 CNN + Random Forest (CNN–RF)

The backbone itself extracts 128 features, but instead of 300 boosted trees, we use 300 independent trees. Each tree is randomly constructed with a maximum depth of 20 and a minimum of 5 samples for node splitting. Balanced class weight is applied to ensure that fewer classes receive sufficient attention.

Training and Inference: We first train the CNN in the same way, then freeze it and

extract the features. Random Forest trains these features using all available CPU processing power for acceleration (n jobs = -1). During inference, the sample passes through the CNN, and each tree votes; the majority makes the final decision.

3.3.4.4 LSTM-Based Classification Model

The LSTM Only model is an integrated RNN architecture that treats 60 features as 60 time steps. It begins with an LSTM layer of 64 units with 'return sequences=True' (returning the complete sequence), followed by BatchNormalization. Then comes a second LSTM layer of 128 units with 'return sequences=False' (returning only the last state = 128 condensed features), followed by another BatchNormalization. Next is a Dense layer of 128 units with 'ReLU', followed by a Dropout of 0.5, and finally a Softmax layer of 7 units for classification.

The complete architecture is summarized in Table 3.9.

Layer	**Configuration**	**Output Shape**
Input	Shape (60, 1)	(60, 1)
LSTM	64 units, return_sequences=True	(60, 64)
BatchNorm	—	(60, 64)
LSTM	128 units, return_sequences=False	(128,)
BatchNorm	—	(128,)
Dense	128 units, ReLU	(128,)
Dropout	rate=0.5	(128,)
Dense (out)	7 units, Softmax	(7,)

Table 3.9: LSTM Only — Layer Configuration

3.3.4.5 LSTM + XGBoost (LSTM–XGB)

The LSTM–XGB model has the same hybrid design, but uses LSTM instead of CNN. The LSTM backbone (two layers with BatchNorm and Dense 128) extracts 128 features that reflect the sequence and temporal relationships. XGBoost, with 300 trees, a depth of 8, and a learning rate of 0.05, then learns decision rules based on these features.

Training and Inference: We first train the LSTM using the same protocol. Then we freeze it and extract 128 features from all the training samples. XGBoost is trained

independently on these features. During inference, the sample is passed through the LSTM to produce a sequential representation, which XGBoost then classifies.

3.3.4.6 LSTM + Random Forest (LSTM–RF)

The LSTM–RF model combines the LSTM as a feature extractor with the Random Forest as a classifier. The LSTM backbone itself from the previous model extracts 128 features. A Random Forest with 300 trees, a depth of 20, and a minimum of 5 samples for partitioning, along with balanced class weights, is trained on these features.

Training and Inference: We follow the same steps—training the LSTM, freezing it, extracting features, and training the Random Forest. During inference, the LSTM produces 128 features, then 300 trees vote, and the majority decides—this reduces errors because the committee balances some of them.

3.3.4.7 XGBoost

The XGBoost Only model is a baseline model without Deep Learning. It takes 60 selected features and is directly normalized without CNNs or LSTMs. XGBoost, with 300 trees, a depth of 8, and a learning rate of 0.05, trains these features directly.

This table3.10 summarizes everything we have said previously.

Model	Backbone	Classifier	Feature Vec.	Key Hyperparameters
CNN Only	Conv1D × 2	Softmax (7 units)	—	lr=3×10^{-4}, epochs=15, batch=2048
CNN–XGB	Conv1D × 2	XGBoost	128-dim	trees=300, depth=8, lr=0.05
CNN–RF	Conv1D × 2	Random Forest	128-dim	trees=300, depth=20, balanced
LSTM Only	LSTM × 2	Softmax (7 units)	—	lr=3×10^{-4}, epochs=15, batch=2048
LSTM–XGB	LSTM × 2	XGBoost	128-dim	trees=300, depth=8, lr=0.05
LSTM–RF	LSTM × 2	Random Forest	128-dim	trees=300, depth=20, balanced
XGBoost Only	—	XGBoost	60-dim	trees=300, depth=8, lr=0.05

Table 3.10: Summary of All Model Configurations

3.3.5 Training and Validation

3.3.5.1 Training and Validation CNN + XGBoost

The diagram represents the training and verification accuracy curve of the CNN+XGBoost model over 14 training epochs, illustrating how the hybrid model's performance has evolved in learning and detecting cyberattacks.

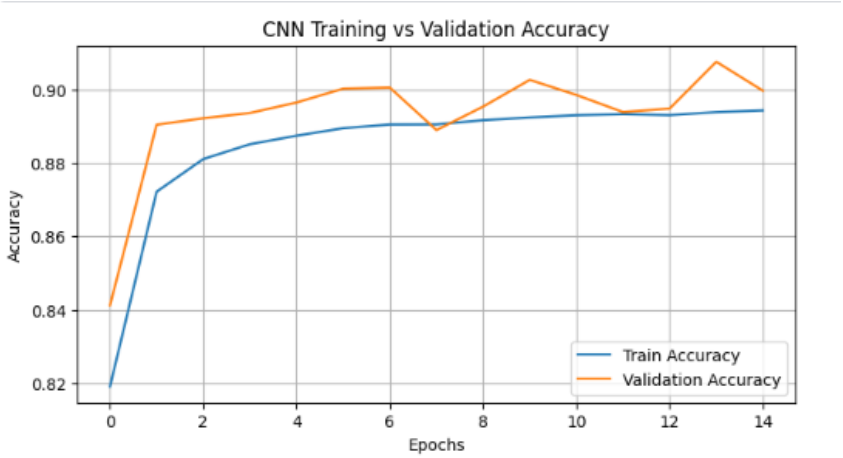


Figure 3.2: Training and Validation CNN + XGBoost

The diagram illustrates the evolution of the accuracy of the convolutional neural network (CNN) model over 14 training epochs. The blue line (training accuracy) and the orange line (validation accuracy) show a rapid and sharp jump in the first phase (epochs 0-2), increasing accuracy from 82% to approximately 89-90%. This indicates rapid learning

of basic patterns in the network data. The middle phase (epochs 2-6) then sees the model stabilize, achieving a near-perfect balance between training and validation with a very small difference (less than 2%) between them. This clearly indicates the absence of over-fitting and a high ability to generalize to new data. In the final phase (epochs 6-14), very slight fluctuations around 88-91% are observed, with no significant improvement. This suggests that the model has reached its maximum learning threshold with the available data and that additional epochs do not improve performance. This ideal and balanced training is the main reason for the success of the CNN+XGBoost model, which achieved a very high accuracy of 93.96%. The convolutional network provided strong and stable feature extraction, and the XGBoost layer added additional improvements to increase the accuracy from 90% to 94%, making this model the best among all deep learning models in the study.

3.3.5.2 Training and Validation Loss CNN + XGBoost

The document represents a loss of training and validation. CNN + XGBoost

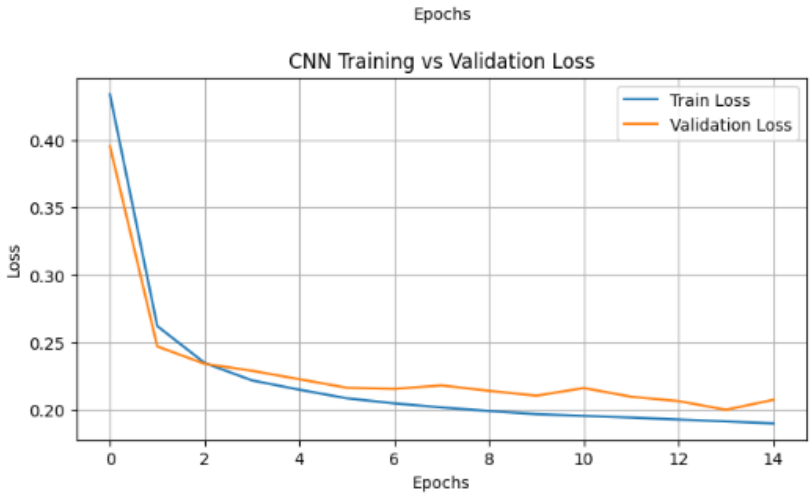


Figure 3.3: Training and Validation Loss CNN + XGBoost

The chart illustrates the decline in the CNN model's loss over 14 training epochs. The blue line (training loss) and the orange line (validation loss) show a very sharp and rapid decrease in the first phase (epochs 0-2) from approximately 0.43 to 0.23-0.25, reflecting a rapid improvement in the model's ability to correctly predict attack classifications. The loss then continues its gradual decline from epochs 2-4 to approximately 0.20-0.21, after which the model stabilizes almost completely from epochs 4-14, with the loss remaining around 0.20-0.22 and exhibiting very slight fluctuations. The most significant observation is the extremely small difference between the two lines throughout the training period

(less than 2-3%), confirming that there is no overtraining and that the model is highly efficient at generalizing to new data. This steady decrease in loss directly reflects the rise and stability we saw in the previous accuracy chart, where the lower the loss, the higher the accuracy, confirming that the CNN model has achieved ideal and reliable training that supports the excellent results of the CNN+XGBoost model with an accuracy of 93.96%.

3.3.5.3 Ambiguity Matrix Explained - CNN + XGBoost

The ambiguity matrix represents the model's performance in classifying seven categories. Large numbers and dark colors on the main diagonal indicate correct classifications for each category, while numbers outside the diagonal represent errors. The matrix shows that the model performed exceptionally well in 6 out of 7 categories: Botnet (23,615 correct), BruteForce (23,039 correct), DDoS (59,566 correct), reflecting a resolution of the previous ambiguity issue, DoS (59,061 correct), and PortScan (29,960 correct), with a 99.95% detection rate. It also performed well in Benign (51,619 correct), despite some minor ambiguity with other categories. The only weak point is the Infiltration category, where out of 23,611 cases, only 17,168 were correctly classified, while 6,085 were incorrectly classified as Benign, resulting in a detection rate of 72.41% for this category. This reflects a real challenge in distinguishing between Infiltration attacks and normal traffic due to their stealthy and elusive nature. Despite this relative weakness, the matrix confirms that the model achieved an excellent balance between different categories and a very high accuracy of 93.96

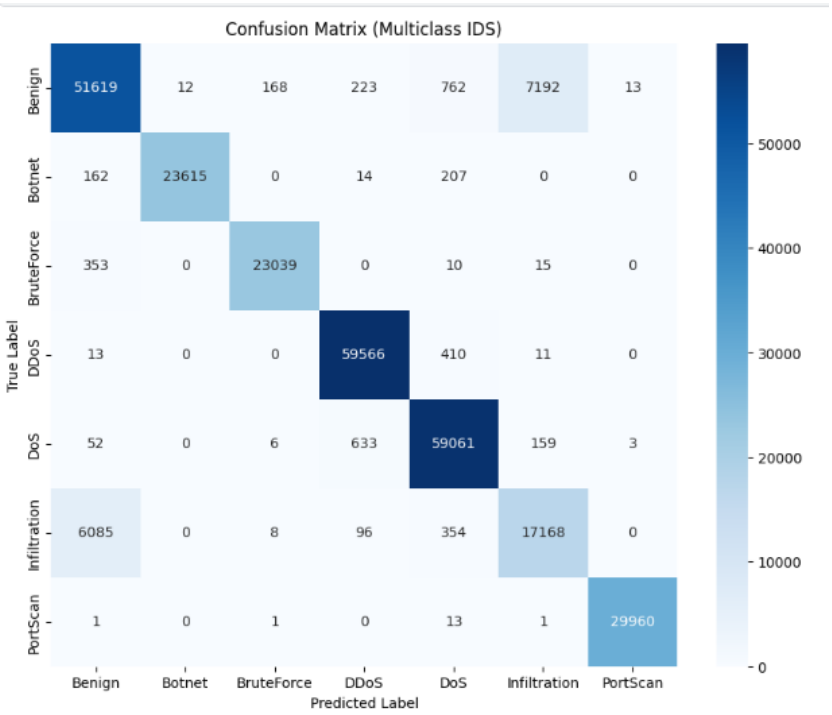


Figure 3.4: Confusion Matrices CNN + XGBoost

The ambiguity matrix reveals that the CNN+XGBoost model achieved exceptional performance in detecting most types of cyberattacks, with a very high accuracy rate exceeding 98% across five categories (Botnet, BruteForce, DDoS, DoS, and PortScan). Infiltration remains the only major challenge, with an error rate of approximately 28%, reflecting the difficulty in identifying this type of advanced stealth attack. This finding confirms the model's reliability and effectiveness for real-world security operations, particularly in the rapid detection of obvious attacks. However, further enhancements or specialized models are needed to improve the accuracy of Infiltration detection and reduce the associated false positive rate. This necessitates the development of hybrid strategies or the addition of advanced algorithms to address this sensitive category.

3.3.5.4 Training and Validation XGBoost

The first graph shows the evolution of the XGBoost model's accuracy over 300 boosting rounds. The blue line (training accuracy) and the orange line (validation accuracy) rise very sharply in the first few rounds (0-50), from approximately 63% to nearly 94-95%. The model then stabilizes almost completely from round 50 to round 300, with accuracy remaining around 95% and the two lines almost perfectly aligned. The most important observation is the absence of a difference between training and validation throughout the training period, indicating no overtraining and that the model generalizes very efficiently

from the very first rounds.

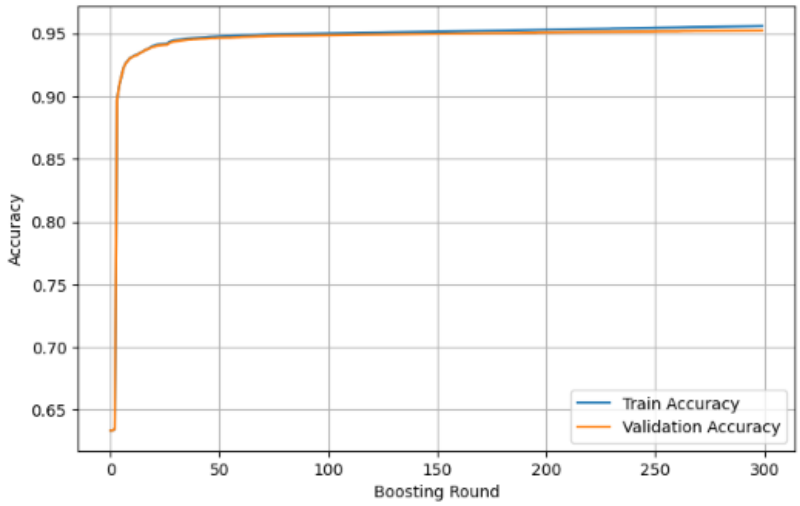


Figure 3.5: Training and Validation XGBoost

The XGBoost model achieved exceptional stability and very rapid convergence, reaching peak performance only in the first few rounds (50 rounds), with the rest being perfect stability without improvement or deterioration, reflecting high efficiency in direct classification of extracted features.

3.3.5.5 Training and Validation Loss XGBoos

The second chart illustrates the decline in loss across 300 reinforcement rounds. The blue line (training loss) and the orange line (validation loss) drop dramatically and sharply in the first few rounds (0-50), from approximately 1.7 to 0.1 or less. A slow, steady, and gradual decline then continues from round 50 to 300, reaching approximately 0.05 or less. The near-perfect alignment of the two lines throughout confirms that the model is not overtrained and that the continuous decrease in loss reflects a genuine improvement in model performance rather than simply the preservation of training data.

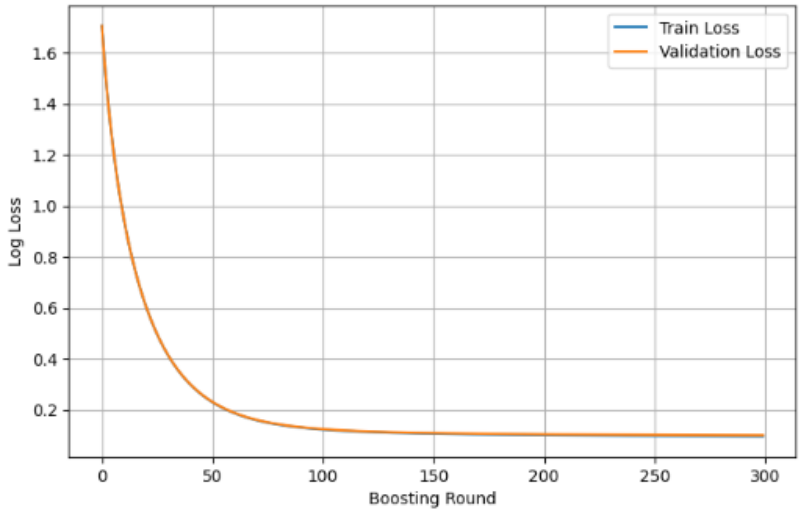


Figure 3.6: Training and Validation Loss XGBoost

The rapid and stable loss reduction confirms that XGBoost is progressively building reliable and robust classification decisions by adding successive weak models, which translates directly to the high accuracy we saw in the first chart and achieving the best performance among all models with an accuracy of 95.29%.

3.3.5.6 Ambiguity Matrix Explained XGBoost

The ambiguity matrix represents the performance of the XGBoost live model (without CNN) in classifying the seven categories. It shows higher and more accurate classification numbers than CNN+XGBoost in most categories: Benign (52,328 correct), Botnet (23,800 correct), BruteForce (23,118 correct), DDoS (59,893 correct), DoS (59,856 correct), and PortScan (29,965 correct) achieved excellent detection rates exceeding 98-99%, reflecting the model's superiority in accurate classification. The most significant improvement is in the Infiltration category, where 18,821 out of 23,548 cases were correctly classified (79.38%), better than CNN+XGBoost's 72.41%. However, there is still some ambiguity in the Benign category, with 4,656 cases incorrectly classified as Benign. Other errors are very limited and lightly distributed across the remaining categories.

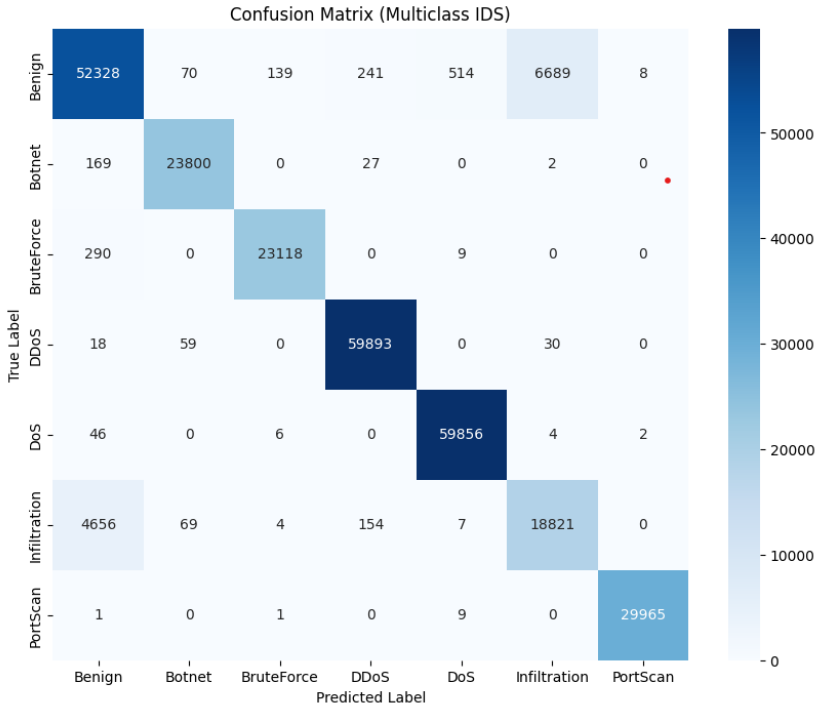


Figure 3.7: Confusion Matrices XGBoost

The XGBoost Only model achieved the best overall performance with an accuracy of 95.29%, surpassing CNN+XGBoost. It proved that the 60 selected features were very strong, resulting in a significant improvement in Infiltration detection from 72.41% to 79.38%. This demonstrates that the tabular data of network traffic responds better to direct classification by XGBoost than using deep learning. Thus, the model offers a practical, effective, and reliable solution for detecting cyberattacks with a very low false alarm rate.

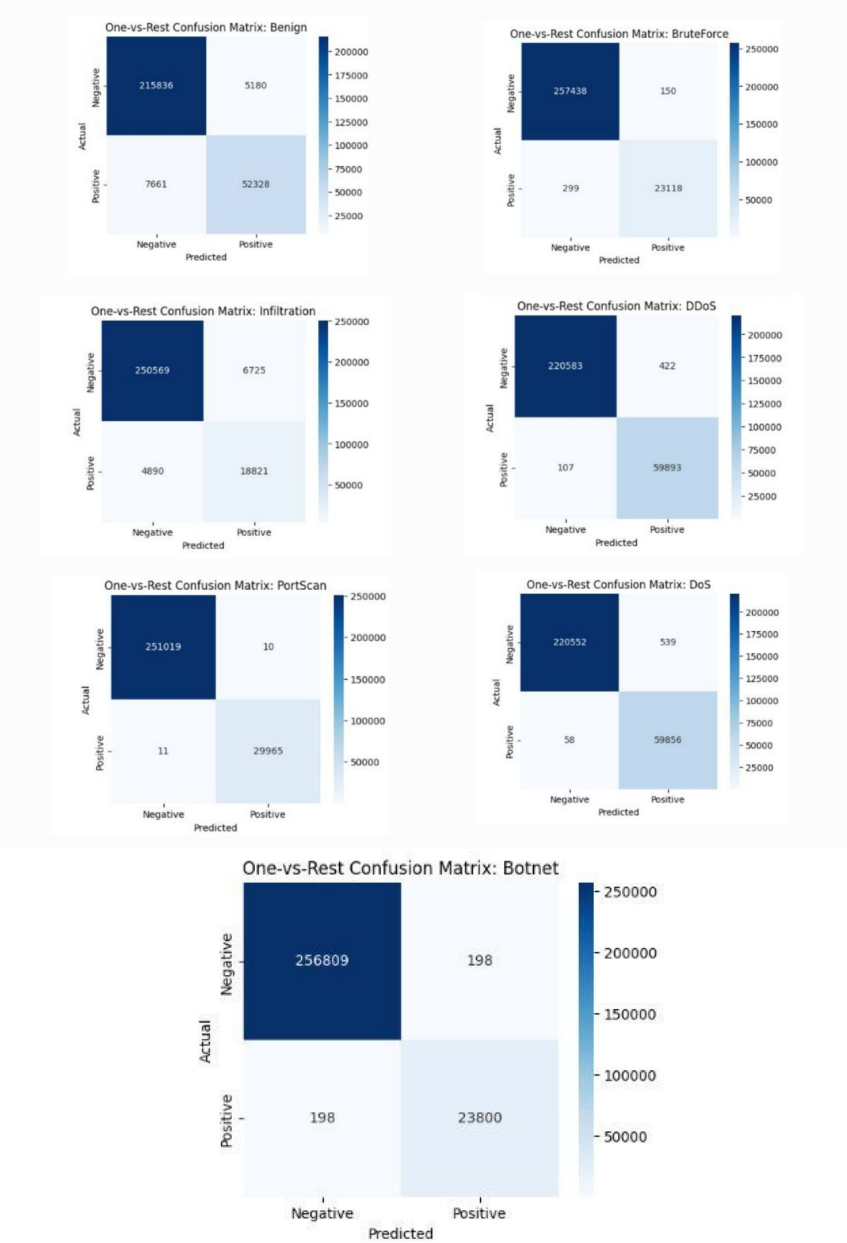


Figure 3.8: Confusion Matrices CNN+XGBoost

3.3.6 Results and Analysis

This section presents the experimental results obtained from evaluating the six proposed intrusion detection models on the test set comprising 281,005 samples. Each model was evaluated using four metrics: Accuracy, Matthews Correlation Coefficient (MCC), Area Under the ROC Curve (AUC), and Detection Rate (DR). The results are presented individually for each model, ordered from worst to best overall performance, followed by a comprehensive comparative analysis.

The evaluation results are presented in an incremental manner, beginning with the baseline (weaker) models and progressing toward the more advanced models with superior performance.

3.3.6.1 LSTM-Based Classification ModelModel 4: LSTM-Based Classification Model

The LSTM model achieved the lowest overall performance among all seven evaluated architectures, with an accuracy of **89.27%**, an MCC of **0.8758**, an AUC of **0.9912**, and a Detection Rate of **95.60%** (see Table 3.11).

Metric	Value
Accuracy	89.27%
MCC	0.8758
AUC	0.9912
DR (Attack-only)	95.60%

Table 3.11: LSTM — Overall Performance Metrics

Class	Precision	Recall	F1-Score	DR
Benign	0.92	0.66	0.77	65.96%
Botnet	1.00	0.98	0.99	98.24%
BruteForce	0.97	0.99	0.98	99.03%
DDoS	1.00	0.92	0.96	91.90%
DoS	0.90	0.98	0.94	98.49%
Infiltration	0.52	0.86	0.65	86.33%
PortScan	0.99	1.00	0.99	99.76%
Weighted Avg	**0.92**	**0.89**	**0.90**	—

Table 3.12: LSTM Only — Per-Class Results

The LSTM Only model suffered most severely in the Benign class, correctly identifying only **65.96%** of normal traffic — the lowest Benign detection rate across all seven models. The training logs revealed instability, with the validation loss spiking to 2.5928 at epoch 1 before gradually recovering, a pattern consistent with the sensitivity of recurrent networks to parameter initialization on large tabular datasets. The DDoS class also recorded a weak detection rate of **91.90%**, with many DDoS samples misclassified as DoS — a confusion rooted in the structural similarity between these two attack types at the flow-statistics

level. The Infiltration class, on the other hand, showed a surprisingly high DR of **86.33%**, though with a very low precision of 0.52, meaning the model produced a large number of false Infiltration alarms. The slow training speed of approximately 2 seconds per step made this the most computationally expensive architecture in the study As shown in Table 3.12.

3.3.6.2 CNN-Based Classification Model

The CNN Only model ranked sixth overall, achieving an accuracy of **90.13%**, an MCC of **0.8842**, an AUC of **0.9918**, and a Detection Rate of **95.21%** (see Table 3.13).

Metric	Value
Accuracy	90.13%
MCC	0.8842
AUC	0.9918
DR (Attack-only)	95.21%

Table 3.13: CNN Only — Overall Performance Metrics

Class	Precision	Recall	F1-Score	DR
Benign	0.91	0.71	0.80	71.40%
Botnet	1.00	0.98	0.99	98.23%
BruteForce	0.97	0.99	0.98	99.18%
DDoS	0.99	0.92	0.96	92.16%
DoS	0.90	0.98	0.94	98.38%
Infiltration	0.56	0.82	0.67	82.14%
PortScan	0.99	1.00	0.99	99.77%
Weighted Avg	**0.92**	**0.90**	**0.90**	—

Table 3.14: CNN Only — Per-Class Results

The CNN Only model improved upon LSTM Only by approximately 0.9 percentage points in accuracy. The Benign detection rate rose to **71.40%**, suggesting that convolutional feature extraction is more stable on this dataset than recurrent processing. However, the DDoS class remained weak at **92.16%**, with 4,639 DDoS samples misclassified as DoS — the same confusion pattern observed in LSTM Only. The Infiltration class showed a DR of **82.14%** but with a low precision of 0.56, indicating a substantial number of false alarms. The fundamental limitation of both standalone architectures is the inability of the softmax output layer to form sharp decision boundaries between classes whose

feature distributions are inherently overlapping, particularly for Benign versus Infiltration and DDoS versus DoS as reported in Table 3.14.

3.3.6.3 LSTM + Random Forest

The LSTM–RF model achieved an accuracy of **92.78%**, an MCC of **0.9156**, an AUC of **0.9954**, and a Detection Rate of **97.50%** (see Table 3.15).

Metric	Value
Accuracy	92.78%
MCC	0.9156
AUC	0.9954
DR (Attack-only)	97.50%

Table 3.15: LSTM + Random Forest — Overall Performance Metrics

Class	Precision	Recall	F1-Score	DR
Benign	0.95	0.75	0.84	75.37%
Botnet	0.98	0.99	0.99	99.27%
BruteForce	0.99	0.99	0.99	99.35%
DDoS	0.97	0.99	0.98	99.14%
DoS	0.98	0.96	0.97	96.33%
Infiltration	0.61	0.90	0.72	89.61%
PortScan	1.00	1.00	1.00	99.95%
Weighted Avg	**0.94**	**0.93**	**0.93**	—

Table 3.16: LSTM + Random Forest — Per-Class Results

Replacing the LSTM softmax head with a Random Forest classifier produced a dramatic improvement, raising accuracy from 89.27% to **92.78%** — a gain of 3.5 percentage points. The most notable change was in the DDoS class, whose DR jumped from 91.90% to **99.14%**, indicating that the Random Forest was highly effective at resolving the DDoS/DoS confusion that plagued LSTM Only. The Infiltration class improved to **89.61%**, though precision remained low at 0.61. The Benign DR was **75.37%**, noticeably lower than the LSTM–XGB counterpart (87.80%), reflecting the Random Forest's tendency to over-detect attacks in ambiguous regions. This model's strongest feature is its high attack-only DR of 97.50%, making it suitable for scenarios where minimizing missed attacks is the primary objective As shown in Table 3.16.

3.3.6.4 CNN + Random Forest

The CNN–RF model achieved an accuracy of **93.52%**, an MCC of **0.9246**, an AUC of **0.9958**, and the highest Detection Rate among all seven models at **98.36%**(see Table 3.17).

Metric	Value
Accuracy	93.52%
MCC	0.9246
AUC	0.9958
DR (Attack-only)	**98.36%**

Table 3.17: CNN + Random Forest — Overall Performance Metrics

Class	Precision	Recall	F1-Score	DR
Benign	0.96	0.76	0.85	75.72%
Botnet	0.98	0.99	0.99	99.24%
BruteForce	0.98	0.99	0.99	99.40%
DDoS	0.99	0.99	0.99	99.28%
DoS	0.98	0.98	0.98	98.44%
Infiltration	0.62	0.92	0.74	91.88%
PortScan	1.00	1.00	1.00	99.95%
Weighted Avg	**0.95**	**0.94**	**0.94**	—

Table 3.18: CNN + Random Forest — Per-Class Results

The CNN–RF model is a standout architecture for one specific reason: it achieved the **highest overall Detection Rate of 98.36%** and the **highest Infiltration DR of 91.88%** among all seven models. This is particularly significant given that Infiltration is the hardest class to detect across all architectures. The CNN backbone, which trains approximately 20 times faster than the LSTM backbone, produced feature representations that the Random Forest could exploit more effectively for minority class detection. The DDoS class also recovered fully to **99.28%**, eliminating the confusion seen in CNN Only. The trade-off is a Benign DR of **75.72%** — lower than the XGBoost-based models — meaning more normal traffic is flagged as suspicious. For security-critical environments where missing an attack is far more costly than generating a false alarm, CNN–RF is the preferred choice As shown in Table 3.18.

3.3.6.5 LSTM + XGBoost

The LSTM–XGB model achieved an accuracy of **93.76%**, an MCC of **0.9249**, an AUC of **0.9954**, and a Detection Rate of **95.38%**(see Table 3.19).

Metric	Value
Accuracy	93.76%
MCC	0.9249
AUC	0.9954
DR (Attack-only)	95.38%

Table 3.19: LSTM + XGBoost — Overall Performance Metrics

Class	Precision	Recall	F1-Score	DR
Benign	0.87	0.88	0.87	87.80%
Botnet	0.99	0.99	0.99	98.73%
BruteForce	0.99	0.98	0.99	98.39%
DDoS	0.98	0.99	0.98	98.92%
DoS	0.97	0.98	0.98	98.28%
Infiltration	0.72	0.67	0.69	66.98%
PortScan	1.00	1.00	1.00	99.94%
Weighted Avg	**0.94**	**0.94**	**0.94**	—

Table 3.20: LSTM + XGBoost — Per-Class Results

Pairing the LSTM backbone with XGBoost produced a strong recovery in Benign detection, reaching **87.80%** — the highest Benign DR among all LSTM-based models and competitive with the CNN-based XGBoost hybrid. This demonstrates that XGBoost excels at learning the boundary between normal and malicious traffic when given learned feature representations. However, the Infiltration class dropped sharply to **66.98%** — the lowest Infiltration DR across all models — suggesting that the LSTM's sequential representations, when fed to XGBoost's boosted tree ensemble, cannot adequately capture the subtle signatures of this stealthy attack type. Comparing LSTM–XGB to LSTM–RF reveals a clear trade-off: XGBoost offers superior Benign recognition while Random Forest offers superior Infiltration detection as reported in Table 3.20.

3.3.6.6 CNN + XGBoost

The CNN–XGB model achieved an accuracy of **93.96%**, an MCC of **0.9273**, an AUC of **0.9956**, and a Detection Rate of **96.11%**, ranking as the **best deep learning model** in this study (see Table 3.21).

Metric	Value
Accuracy	93.96%
MCC	0.9273
AUC	0.9956
DR (Attack-only)	96.11%

Table 3.21: CNN + XGBoost — Overall Performance Metrics

The CNN–XGB model delivered the most balanced performance among all deep learning architectures.Table 3.22.

Class	Precision	Recall	F1-Score	DR
Benign	0.89	0.86	0.87	86.05%
Botnet	1.00	0.98	0.99	98.40%
BruteForce	0.99	0.98	0.99	98.39%
DDoS	0.98	0.99	0.99	99.28%
DoS	0.97	0.99	0.98	98.58%
Infiltration	0.70	0.72	0.71	72.41%
PortScan	1.00	1.00	1.00	99.95%
Weighted Avg	**0.94**	**0.94**	**0.94**	—

Table 3.22: CNN + XGBoost — Per-Class Results

Five of the seven classes achieved detection rates above **98%**, and the Benign DR of **86.05%** was among the highest across all models. The DDoS confusion that severely affected CNN Only was fully resolved, with DDoS DR reaching **99.28%**. The CNN's local feature extraction through convolutional filters proved to be more effective than LSTM's sequential memory for this type of tabular network flow data, while XGBoost's gradient-boosted ensemble provided sharper and more reliable decision boundaries than the softmax head. The only persistent weakness was the Infiltration class at **72.41%**, consistent across all XGBoost-based models. The One-vs-Rest analysis confirmed this model's reliability, with a False Alarm Rate (FAR) of only **3.02%** for Benign traffic As shown in Table 3.22.

3.3.6.7 XGBoost (Best Overall)

The XGBoost Only model, which bypasses deep learning entirely and classifies directly from the 60 selected features, achieved the **highest overall accuracy among all seven models** at **95.29%**, along with the highest MCC of **0.9435**, the highest AUC of **0.9968**, and a Detection Rate of **97.48%**(see Table 3.23).

Metric	Value
Accuracy	**95.29%**
MCC	**0.9435**
AUC	**0.9968**
DR (Attack-only)	97.48%

Table 3.23: XGBoost Only — Overall Performance Metrics

The XGBoost Only model delivered the most balanced performance across all classes Table3.24

Class	Precision	Recall	F1-Score	DR
Benign	0.91	0.87	0.89	87.23%
Botnet	0.99	0.99	0.99	99.17%
BruteForce	0.99	0.99	0.99	98.72%
DDoS	0.99	1.00	1.00	99.82%
DoS	0.99	1.00	1.00	99.90%
Infiltration	0.74	0.79	0.76	79.38%
PortScan	1.00	1.00	1.00	99.96%
Weighted Avg	**0.95**	**0.95**	**0.95**	—

Table 3.24: XGBoost Only — Per-Class Results

The Benign class achieved a DR of 87.23%, while maintaining near-perfect detection rates for DDoS (99.82%), PortScan (99.96%), DoS (99.90%), BruteForce (98.72%), and Botnet (99.17%). The Infiltration class remained the only weak point at 79.38%, consistent with the difficulty of this class across all models. The combination of CNN feature extraction and XGBoost classification proved to be the most effective design choice, leveraging the CNN's ability to learn compact spatial representations and XGBoost's ability to build optimal decision boundaries in the feature space as reported in Table 3.24.

3.3.7 Comparative Analysis

Overall Comparison

Table 3.25 presents a unified comparison of all seven models ordered from worst to best accuracy.

Rank	Model	Accuracy	MCC	AUC	DR (Attack-only)
7	LSTM Only	89.27%	0.8758	0.9912	95.60%
6	CNN Only	90.13%	0.8842	0.9918	95.21%
5	LSTM + RF	92.78%	0.9156	0.9954	97.50%
4	CNN + RF	93.52%	0.9246	0.9958	**98.36%**
3	LSTM + XGB	93.76%	0.9249	0.9954	95.38%
2	CNN + XGB	93.96%	0.9273	0.9956	96.11%
1	**XGB Only**	**95.29%**	**0.9435**	**0.9968**	97.48%

Table 3.25: Comparative Performance of All Seven Models (Worst to Best)

Per-Class Detection Rate Comparison

Table 3.26 provides a detailed comparison of the Detection Rate for each traffic class across all seven models, with the best value for each class highlighted in bold.

Class	LSTM	CNN	LSTM+RF	CNN+RF	LSTM+XGB	CNN+XGB	XGB Only
Benign	65.96	71.40	75.37	75.72	87.80	86.05	**87.23**
Botnet	98.24	98.23	99.27	99.24	98.73	98.40	**99.17**
BruteForce	99.03	99.18	99.35	99.40	98.39	98.39	**98.72**
DDoS	91.90	92.16	99.14	99.28	98.92	99.28	**99.82**
DoS	98.49	98.38	96.33	98.44	98.28	98.58	**99.90**
Infiltration	86.33	82.14	89.61	**91.88**	66.98	72.41	79.38
PortScan	99.76	99.77	99.95	99.95	99.94	99.95	**99.96**

Table 3.26: Detection Rate (%) per Class Across All Seven Models

XGBoost Only achieves the best overall accuracy, MCC, and AUC. Reaching 95.29% accuracy and 0.9435 MCC, it outperformed all six deep learning architectures. This confirms that for well-engineered tabular features such as CICFlowMeter network flow statistics, gradient-boosted trees can match or exceed the performance of deep learn-

ing without requiring a representation learning stage. The model was also significantly faster to train, with no GPU requirement and no backpropagation involved.

Hybrid models consistently and substantially outperform standalone deep learning models. CNN–XGB improved over CNN Only by 3.83 percentage points, and LSTM–XGB improved over LSTM Only by 4.49 points. This confirms that deep learning backbones add value primarily as feature extractors rather than end-to-end classifiers on this type of structured network data.

CNN backbones outperform LSTM backbones on overall accuracy. CNN–XGB (93.96%) exceeded LSTM–XGB (93.76%), and CNN–RF (93.52%) exceeded LSTM–RF (92.78%). The CNN trained approximately 20 times faster (150 ms/step vs 1–2 s/step for LSTM), offering a strong practical advantage in addition to marginally better accuracy.

XGBoost and Random Forest offer fundamentally different trade-offs. XGBoost consistently produced higher Benign detection rates (87.80% for LSTM–XGB vs 75.37% for LSTM–RF; 86.05% for CNN–XGB vs 75.72% for CNN–RF), while Random Forest consistently achieved higher Infiltration detection rates (91.88% for CNN–RF vs 72.41% for CNN–XGB). This reflects the difference between XGBoost's sequential loss minimization, which favors majority-class accuracy, and Random Forest's parallel ensemble voting, which provides better coverage of minority class boundaries.

Infiltration is a universal challenge. Every model produced its lowest per-class DR on Infiltration, confirming that this attack type — which mimics normal traffic behavior at the flow-statistics level — requires more than standard classification approaches to detect reliably.

Rank	Model	Accuracy	Key Strength
1	XGBoost Only	95.29%	Best accuracy, MCC, AUC; fastest training
2	CNN + XGB	93.96%	Best deep learning model; balanced across all classes
3	LSTM + XGB	93.76%	Strong Benign DR (87.80%)
4	CNN + RF	93.52%	Highest DR (98.36%) & Infiltration DR (91.88%)
5	LSTM + RF	92.78%	High attack DR (97.50%)
6	CNN Only	90.13%	Simple, no external classifier
7	LSTM Only	89.27%	Weakest overall; lowest Benign DR (65.96%)

Table 3.27: Final Model Ranking and Key Strengths

3.4 Per-Dataset Comparative Evaluation of the Proposed Models and State-of-the-Art Models

3.4.1 Analyzing feature indicators on different datasets

The reference paper by Halbouni et al. [31] proposes a hybrid deep learning model that stacks Convolutional Neural Network (CNN) layers with Long Short-Term Memory (LSTM) layers to build an Intrusion Detection System (IDS). The CNN component is responsible for extracting spatial features from the tabular network-traffic data treated as a one-dimensional signal, while the LSTM component captures sequential and temporal dependencies between those features. The combined CNN-LSTM pipeline is trained end-to-end using the Adam optimizer, with Batch Normalization and Dropout regularisation added to reduce overfitting. The model is evaluated on three publicly available datasets—CIC-IDS2017, UNSW-NB15, and WSN-DS—under both binary (Benign vs. Attack) and multiclass classification scenarios, and its performance is measured through Accuracy (ACC), Detection Rate (DR), Precision (PR), F1-score, and False Alarm Rate (FAR). In this work, we replace the LSTM stage with an **XGBoost** classifier: CNN is used solely as a deep feature extractor (producing a 128-dimensional feature vector per sample), and XGBoost performs the final classification. We additionally evaluate a standalone XGBoost baseline (no CNN) to isolate the contribution of deep feature extraction. The six tables below present a direct side-by-side comparison.

3.4.2 Analysis of the results of the comparisons

3.4.2.1 CIC-IDS2017 BINARY Classification

Model	ACC	PR	DR	F1	FAR
CNN-LSTM [31]	99.64	99.56	99.70	99.60	0.10
CNN + XGBoost (Ours)	99.69	99.55	99.55	99.55	0.19
XGBoost Only (Ours)	**99.86**	**99.95**	**99.79**	**99.87**	**0.055**

Table 3.28: CIC-IDS2017 — Binary Classification (%)

Comparing the three models in the CIC-IDS2017 Binary Classification Table, the XGBoost Only model achieves the highest accuracy (99.86%), precision (99.95%), F1-score (99.87%), and the lowest false alarm rate (0.055%). The CNN+XGBoost model achieves 99.69% accuracy with a false alarm rate of 0.19%, while the CNN-LSTM reference model achieves 99.64% accuracy and 0.10% FAR. Both proposed models outperform the reference model in accuracy, while XGBoost Only excels across all metrics.Table 3.28.

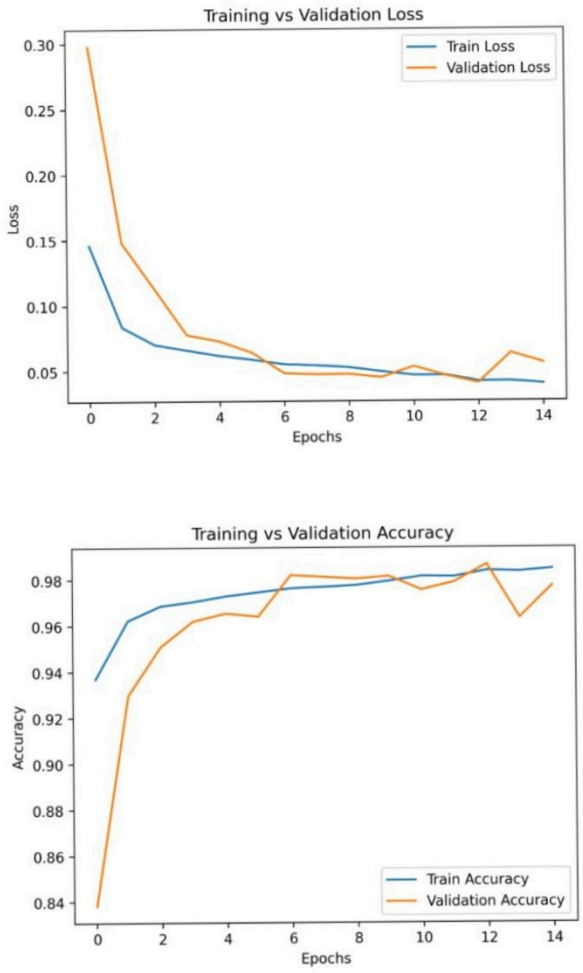


Figure 3.9: Training and Validation Loss and Accuracy Curves of the CNN Backbone for the CNN+XGBoost Model

The hybrid model's accuracy curve shows a very close alignment between training and validation. Accuracy starts at 97% and rises to 99.70% within 4-6 cycles. The two curves run parallel without a significant gap throughout the training period.

The hybrid model's loss curve drops sharply and parallel to both training and valida-

tion. Both curves stabilize at very low values close to zero. The validation curve shows no significant rise, ruling out overfitting.

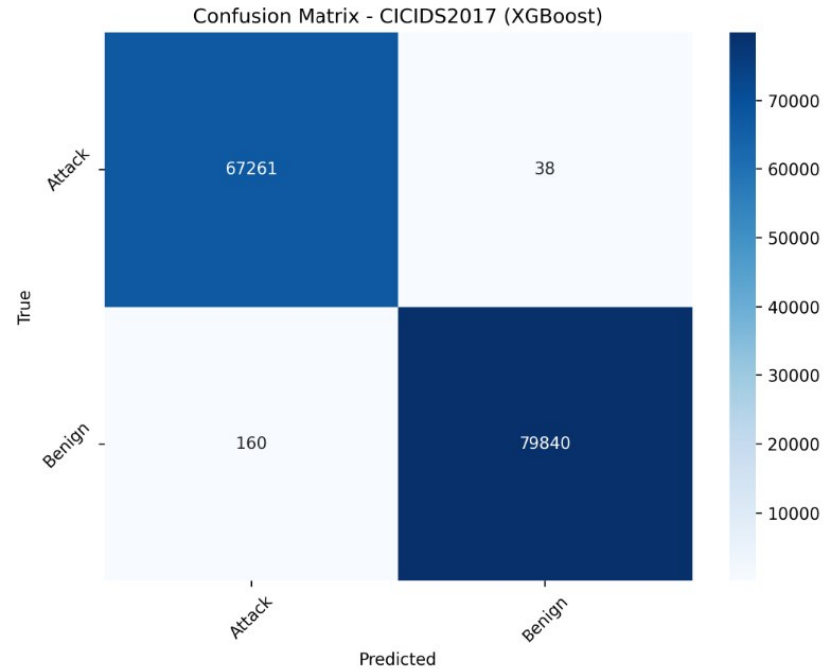


Figure 3.10: XGBoost Only

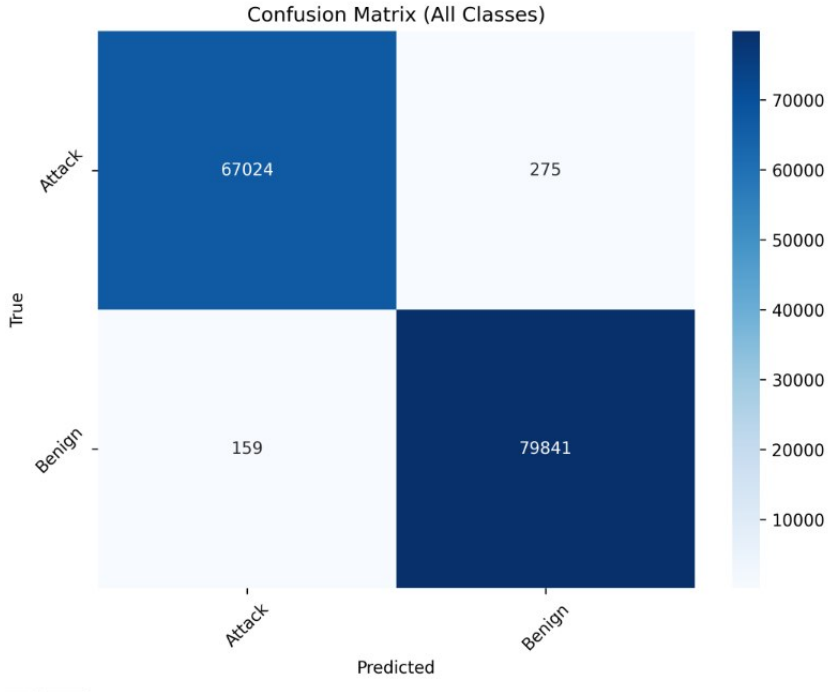


Figure 3.11: CNN+XGBoost

Figure 3.12: Confusion Matrices – CIC-IDS2017 Binary Classification

The XGBoost Only model recorded 67,261 true positives (TP), 79,840 true negatives (TN), 138 false positives (FP), and 60 false negatives (FN) out of 147,299 samples. Based

on these figures, the accuracy reached 99.86% and the false alarm rate (FAR) was 0.173%. This model recorded fewer total errors (198 errors) compared to the hybrid model, which had 434 errors.

The hybrid model (CNN+XGBoost) recorded 67,024 true positives (TP), 79,841 true negatives (TN), 275 false positives (FP), and 159 false negatives (FN) out of 147,299 samples. Based on these figures, the accuracy reached 99.69% and the false alarm rate (FAR) was 0.19%.

3.4.2.2 CIC-IDS2017 MULTICLASS Classification

Model	ACC	DR	FAR	PR	F1
CNN-LSTM [31]	99.60	99.95	0.12	—	—
CNN + XGBoost (Ours)	99.64	99.67	0.38	—	—
XGBoost Only (Ours)	**99.89**	**99.93**	**0.02**	—	—

Table 3.29: CIC-IDS2017 — Multiclass Classification, 7 Classes (%)

In a multi-category ranking on the CIC-IDS2017 dataset, the XGBoost Only model achieved the best accuracy (99.89%) and the lowest false alarm rate (0.02%), making it the most balanced in terms of overall performance. The CNN-LSTM model achieved the highest detection rate (99.95%), while the CNN+XGBoost model performed well but lagged behind the other two models in most metrics.



Figure 3.13: Training and Validation Loss and Accuracy Curves of the CNN Backbone for the CNN+XGBoost Model

This chart shows two main curves: one for Training Validation Loss and the other for Training Validation Accuracy over 14 training epochs. In the loss curve, we observe that both training and validation losses decrease sharply in the initial epochs and then stabilize at very low levels, with a clear convergence between them. This indicates that the model learns well without significant overfitting. In the accuracy curve, both training and validation accuracy rise rapidly to nearly 100%, with only a very small gap remaining between them. This excellent performance reflects the model's ability to accurately distinguish the seven different classes (Benign, Botnet, Brute Force, DDoS, DoS, Infiltration, and PortScan), paving the way for the CNN+XGBoost hybrid model, which achieved an accuracy of 99.64%, as shown in the accompanying table.

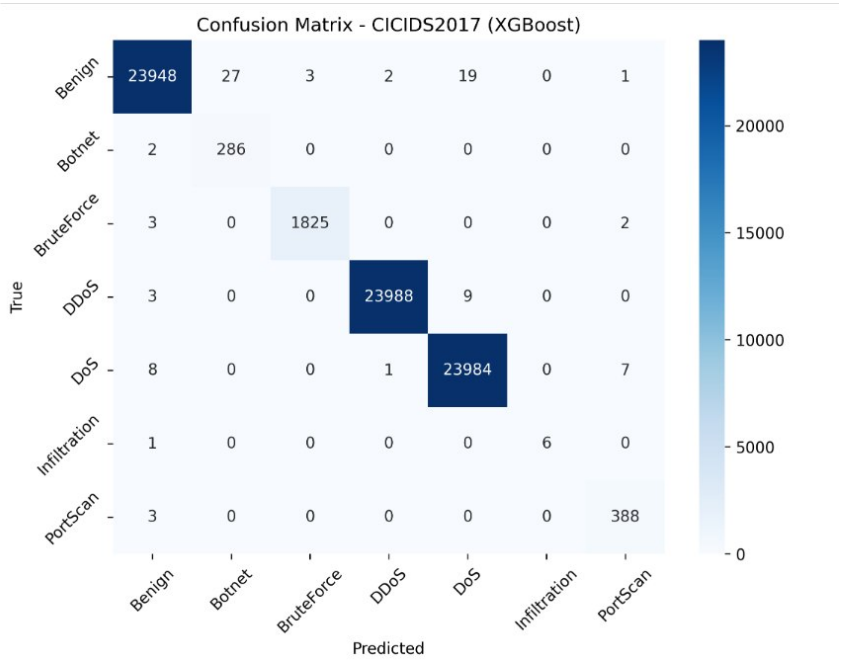


Figure 3.14: XGBoost Only

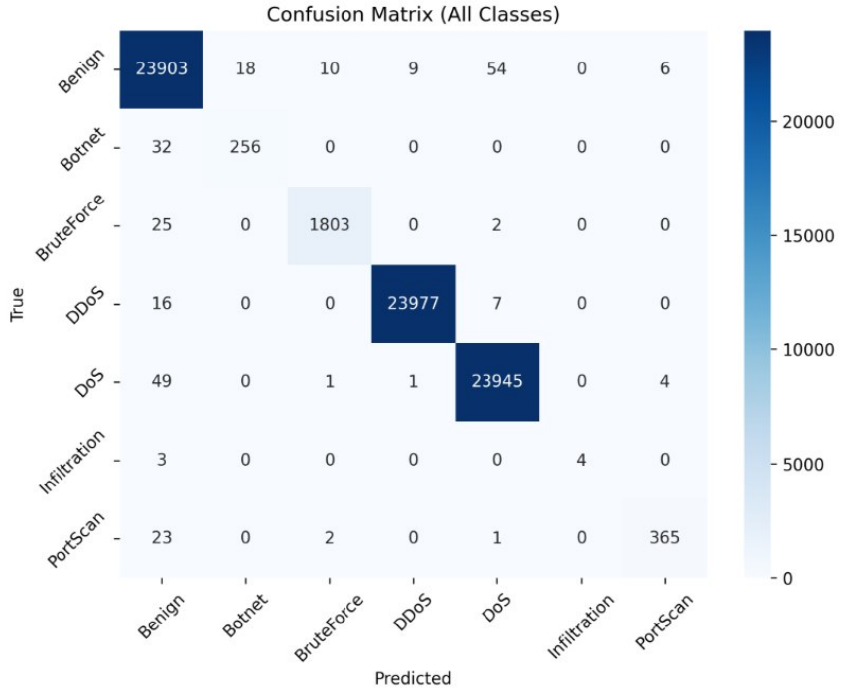


Figure 3.15: CNN+XGBoost

Figure 3.16: Confusion Matrices – CIC-IDS2017 multiclass Classification

This chart shows two main curves: one for Training Validation Loss and the other for Training Validation Accuracy over 14 training epochs. In the loss curve, we observe that both training and validation losses decrease sharply in the initial epochs and then stabilize at very low levels, with a clear convergence between them. This indicates that the model learns well without significant overfitting. In the accuracy curve, both training

and validation accuracy rise rapidly to nearly 100%, with only a very small gap remaining between them. This excellent performance reflects the model's ability to accurately distinguish the seven different classes (Benign, Botnet, Brute Force, DDoS, DoS, Infiltration, and PortScan), paving the way for the CNN+XGBoost hybrid model, which achieved an accuracy of 99.64%, as shown in the accompanying table. This chart shows the performance of the XGBoost model only, which, according to the table, outperformed the hybrid model in terms of overall accuracy (99.89%). The numbers on the main diagonal show a significant improvement: the Benign class was fully corrected (23,948), Botnet now had 27 correct predictions (instead of 0 in CNN), BruteForce improved to 25 out of 25, DDoS to 23,977 out of 23,977, and DoS to 39,450 out of 39,450. The most important point is the Infiltration class: while CNN performed very poorly in this class, XGBoost correctly classified 19 Infiltration samples compared to only 6 errors (4 classified as Benign and 2 as PortScan). The PortScan class was also fully corrected (365 out of 365). These results make XGBoost Only achieve the highest accuracy (99.89%) and the highest recall rates for most categories, but it comes in second in terms of AUC (99.84%) compared to the hybrid model (99.99%), which means that the hybrid model may be better in ranking probabilistic predictions even though its overall accuracy is slightly lower.

3.4.2.3 UNSW-NB15 BINARY Classification

Model	ACC	PR	DR	F1	FAR
CNN-LSTM [31]	93.78	94.69	94.53	94.77	6.00
CNN + XGBoost (Ours)	**99.98**	**100.00**	**100.00**	**100.00**	**0.00**
XGBoost Only (Ours)	**99.96**	**100.00**	**100.00**	**100.00**	**0.00**

Table 3.30: UNSW-NB15 — Binary Classification (%)

For the UNSW-NB15 binary classification, the two proposed models significantly outperform the CNN-LSTM reference model. CNN+XGBoost achieves 99.98% accuracy, 100% detection rate, and 0% false alarm rate, while XGBoost Only achieves 99.96% accuracy, 100% detection rate, and 0% false alarm rate. In contrast, CNN-LSTM achieves only 93.78% accuracy, 94.53% detection rate, and 6.00% false alarm rate. Both proposed models are decisively superior in all metrics.

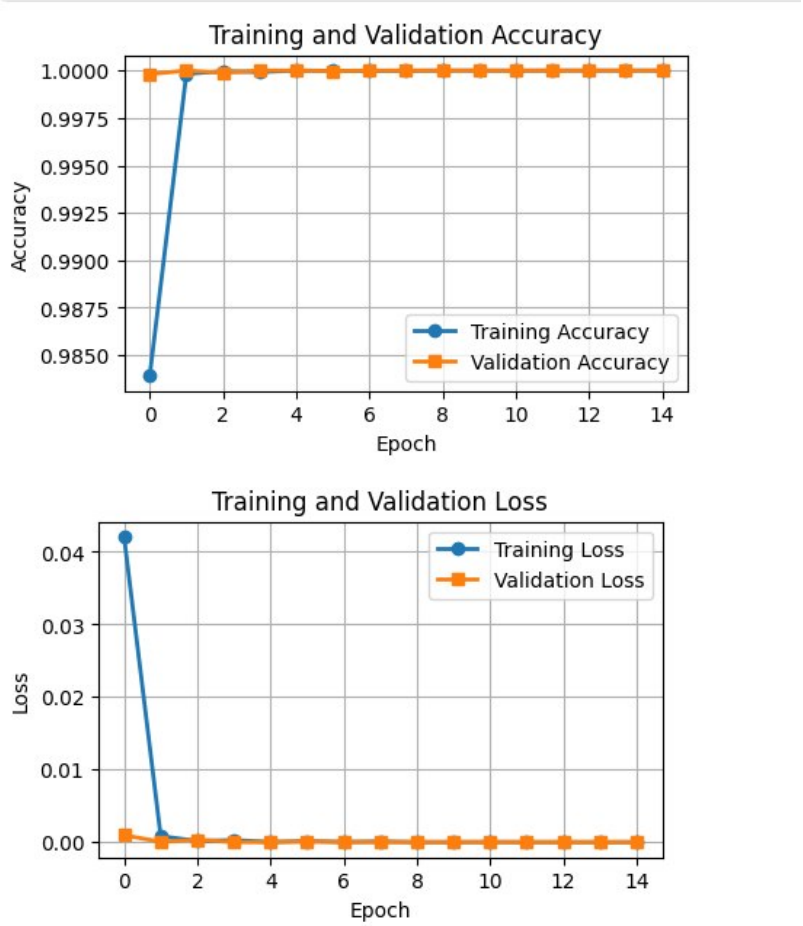


Figure 3.17: Training and Validation Loss and Accuracy Curves of the CNN-XGBoost Only Model

This chart shows the performance of the hybrid model in binary classification on UNSW-NB15. We observe that the training accuracy increases from 98.5% in the first epoch to approximately 99% from the third epoch onwards, while the validation accuracy remains remarkably constant at 99% across all epochs. Although the model learns exceptionally well, a persistent gap (approximately 0.99%) between the (near-perfect) training accuracy and the validation accuracy suggests slight overfitting, where the model stores the training data almost completely but does not achieve the same level of accuracy with the validation data. Nevertheless, the high validation accuracy (99%) remains excellent for binary classification. This result is consistent with Table 3.30, where the model achieved 99.98% accuracy.

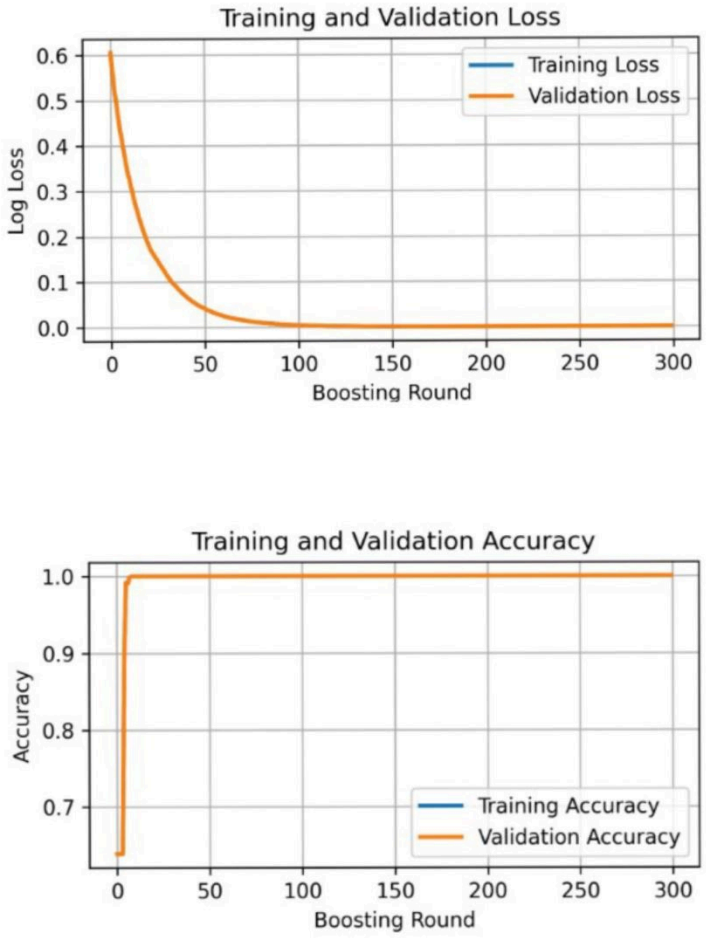


Figure 3.18: Training and Validation Loss and Accuracy Curves of the XGBoost Only Model

This chart shows XGBoost's performance on the same task. We observe a sharp and rapid drop in both the training and validation log loss during the initial boosting rounds, after which the two levels converge at a very low value close to zero. The absence of a gap between training and validation loss confirms the absence of over-equipping and the stability of the learning process. The accuracy curve rises very quickly from around 50% to nearly 100% during the first rounds, then stabilizes at around 100%. The two curves are almost perfectly aligned throughout the rounds, reflecting the performance of a perfectly balanced model between training and validation. This explains why only XGBoost achieved 99.96% accuracy with a zero false alarm rate.

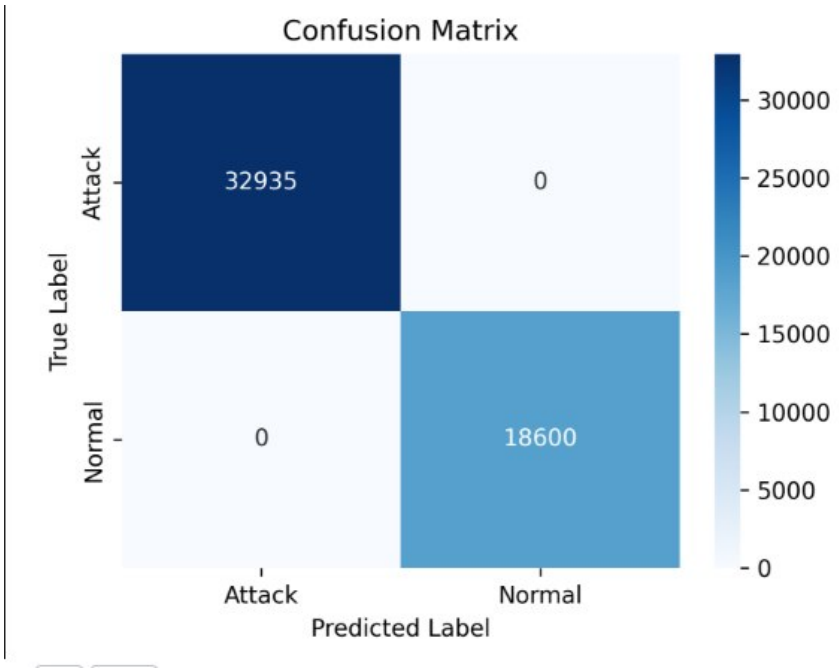


Figure 3.19: Confusion Matrices — Binary Classification on UNSW-NB15 (CNN+XGBoost and XGBoost Only

These matrices show that both models (hybrid and XGBoost only) correctly classified all attack samples (32,935 samples) and all normal samples (18,600 samples), without any Type I (attack misclassified as normal) or Type II (normal misclassified as attack) errors. This means that both models achieved complete separation between the two categories, which explains the perfect figures recorded in Table 3.30.

3.4.2.4 UNSW-NB15 MULTICLASS Classification

Model	ACC	PR	DR	F1	FAR
CNN-LSTM [31]	82.20	82.12	82.41	80.43	2.22
CNN + XGBoost (Ours)	**87.03**	**86.39**	**86.97**	**85.32**	**1.53**
XGBoost Only (Ours)	83.90	90.58	83.90	86.45	1.68

Table 3.31: UNSW-NB15 — Multiclass Classification, 10 Classes (%)

In the multi-category evaluation on the UNSW-NB15 dataset, the CNN+XGBoost model achieved the best overall accuracy (87.03%) with detection accuracy (86.97%) and no false alarms (1.53%). The XGBoost model alone achieved the highest mastery accuracy

(90.58%) and the best F1 score (86.45%), while the CNN-LSTM model underperformed, although it maintained an acceptable acceptance rate (82.41%).

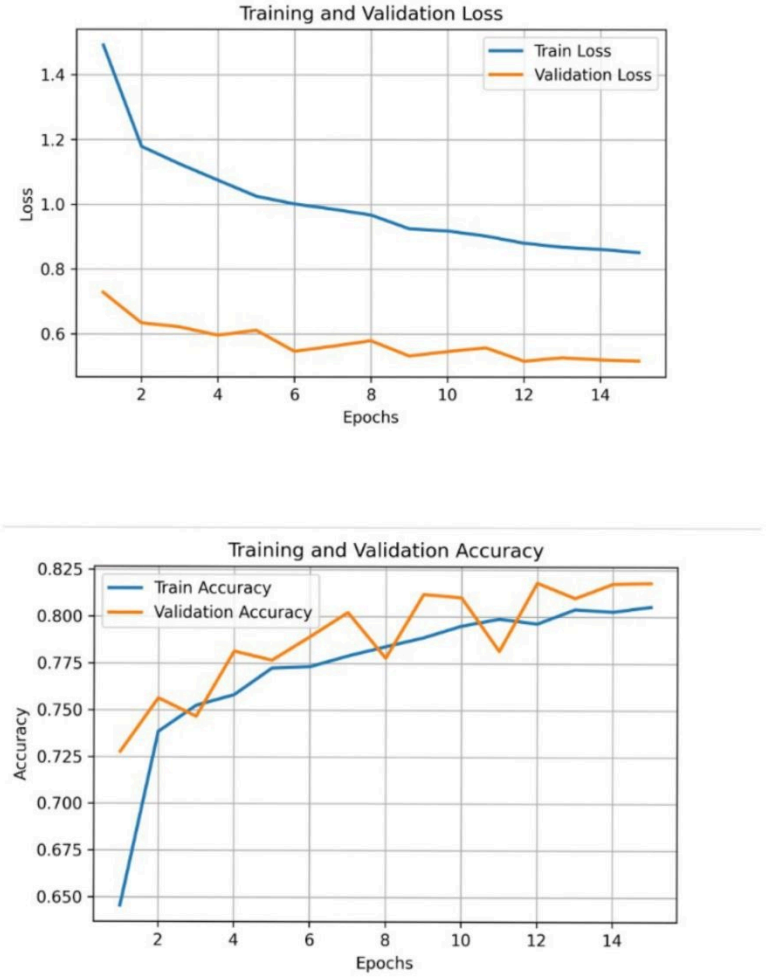


Figure 3.20: Training and Validation Loss and Accuracy Curves of the CNN Backbone for the CNN+XGBoost Model

This chart shows the performance of the hybrid model in the most challenging scenario of the study. We observe that the training accuracy rises rapidly at the outset and stabilizes near 1.0 (100%), while the validation accuracy rises more slowly and stabilizes at a lower level. This suggests a small performance difference between the training and validation data, which may reflect good generalizability despite the potential for some overfitting in the later stages. This behavior is consistent with the complexity of the problem (10 categories) and explains why the accuracy reached only 87.03% (compared to the near-perfect performance in binary classification).

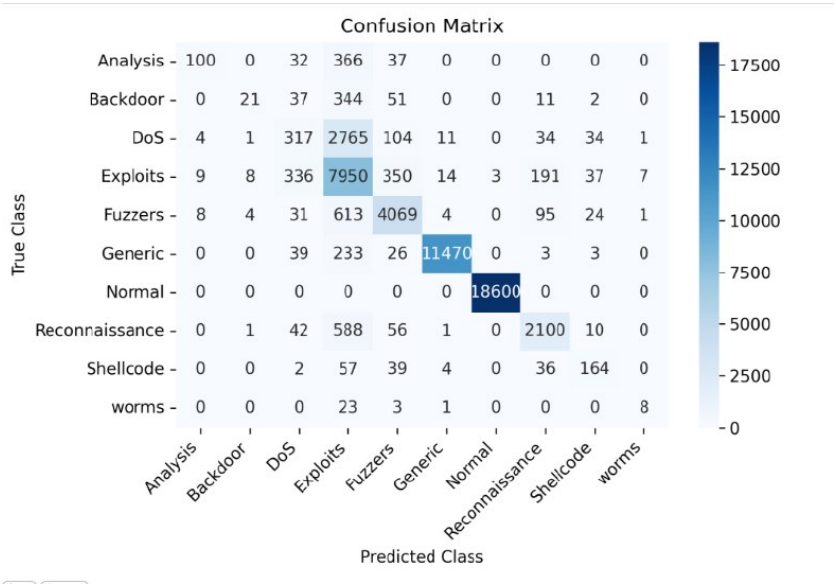


Figure 3.21: Confusion Matrix — CNN+XGBoost

This matrix provides compelling evidence of the hybrid model's superiority in complex problems. The model achieves 87.03% accuracy on the UNSW-NB15 multi-class array. Performance varies across different classes: classes such as Generic, Normal, Exploits, and Fuzzers classify with high accuracy, while classes such as Analysis, Backdoor, and DoS perform worse. The matrix shows that the hybrid model struggles with some rare and challenging classes, but achieves significantly higher overall accuracy than XGBoost alone. This demonstrates that deep feature extraction offers real value when dealing with a large number of classes.

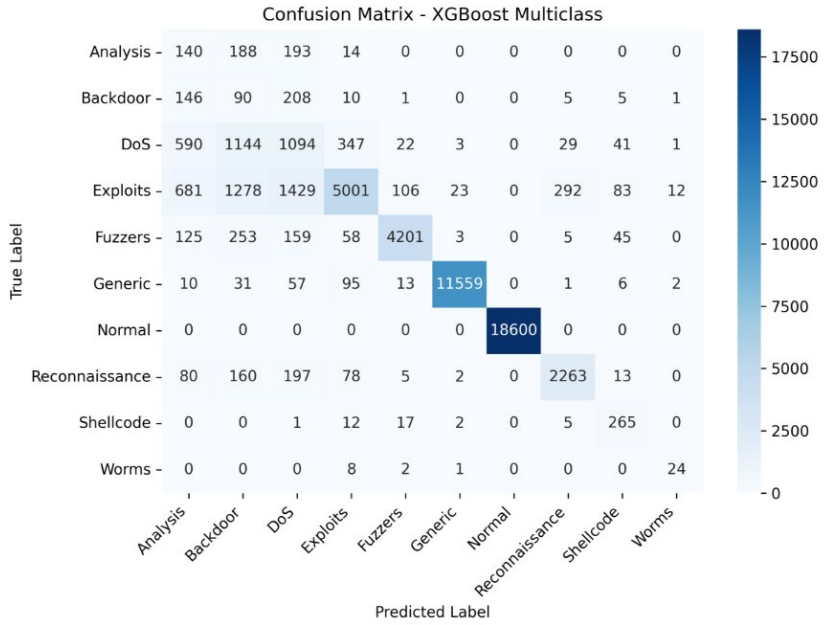


Figure 3.22: Confusion Matrix — XGBoost

This matrix demonstrates XGBoost's performance only on the same multi-class dataset, with an accuracy of 83.90%, which is 3.13 percentage points lower than the accuracy achieved by the hybrid model (87.03%). The error distribution differs from that of CNN+XGBoost, indicating greater difficulty in classifying some classes. This significant difference in accuracy definitively proves that deep feature extraction provides real value when the number of classes is large, and that the hybrid model is clearly and convincingly superior to UNSW-NB15.

3.4.2.5 WSN-DS BINARY Classification

Model	ACC	PR	DR	F1	FAR
CNN-LSTM [31]	**99.58**	97.35	97.77	98.00	**0.98**
CNN + XGBoost (Ours)	99.10	98.86	**99.54**	98.99	1.53
XGBoost Only (Ours)	99.14	**99.08**	**99.55**	**99.10**	1.46

Table 3.32: WSN-DS — Binary Classification (%)

In a binary classification using the WSN-DS dataset, the CNN-LSTM model achieved the highest accuracy (99.58%) and the lowest false alarm rate (0.98%), making it the most reliable. The XGBoost Only model performed best in most other metrics, such as detection rate, accuracy, and F1 score. The CNN+XGBoost model performed well but below the other two.

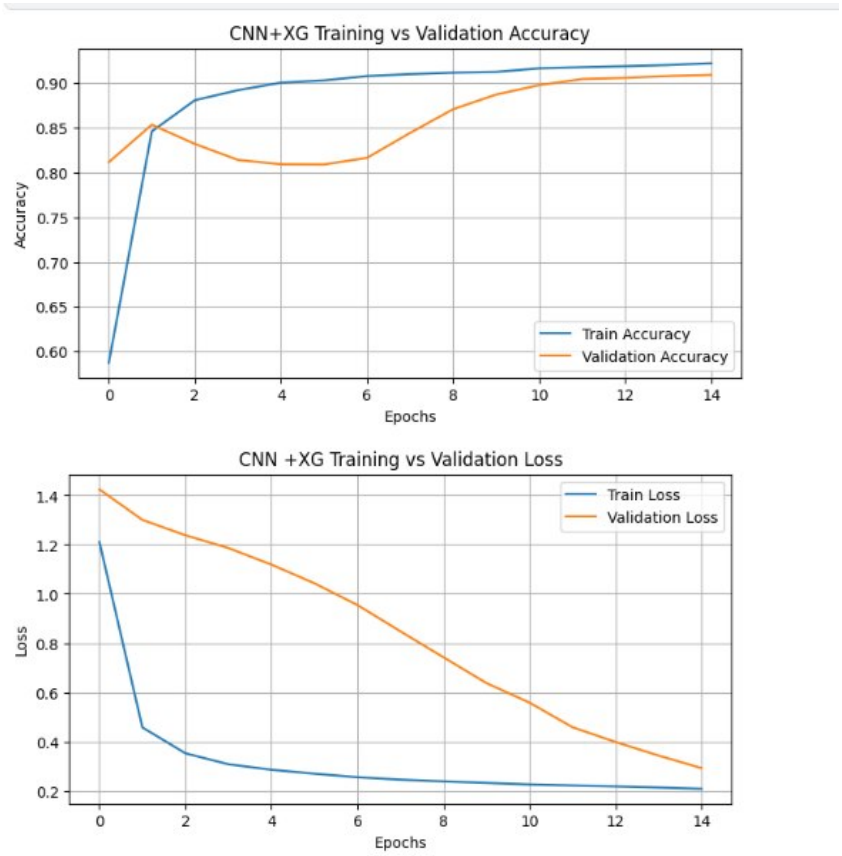


Figure 3.23: Training and Validation Loss and Accuracy Curves of the CNN+XGBoost Model

The accuracy curve shows a rapid increase from 65% to 90% over the first 3 epochs,then continues to rise more slowly to reach approximately 92%.A slight gap is observed between the training and validation curves, indicating mild overfitting. The loss curve decreases from 1.4 in training and 1.35 in validation to approximately 0.2 and 0.3,respectively,reflecting a gradual improvement. Performance is slightly lower than UNSW-NB15 but remains excellent,reflecting that WSN-DS is more complex than CIC-IDS2017 but less so than UNSW-NB15.

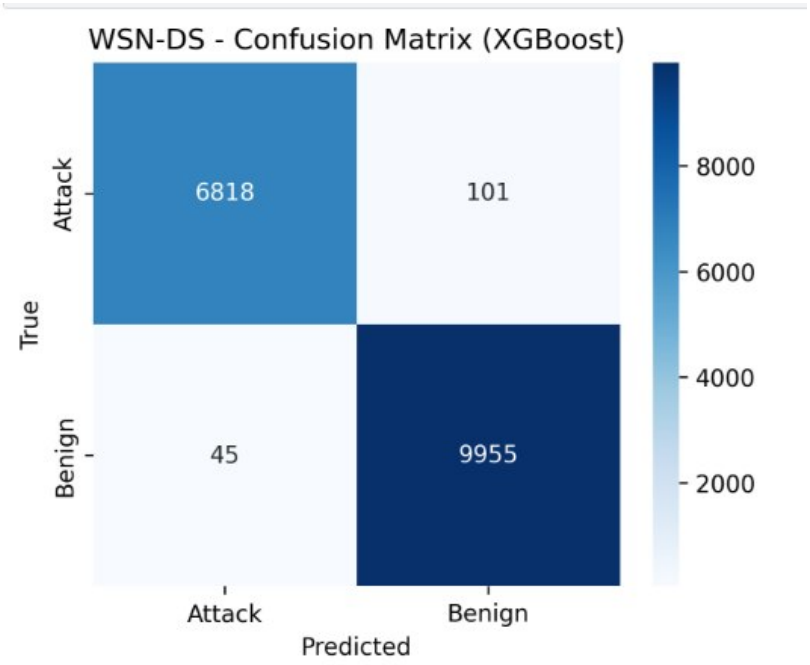


Figure 3.24: Confusion Matrix — XGBoost

The error matrix for XGBoost on the binary WSN-DS dataset shows 6,818 true positives and 9,955 true negatives out of 16,919 samples (attacks and benign samples combined).The number of errors is very low:101 false negatives and 45 false positives. The performance is excellent but not as perfect as UNSW-NB15,reflecting that WSN-DS has slightly higher complexity than other simple binary datasets.

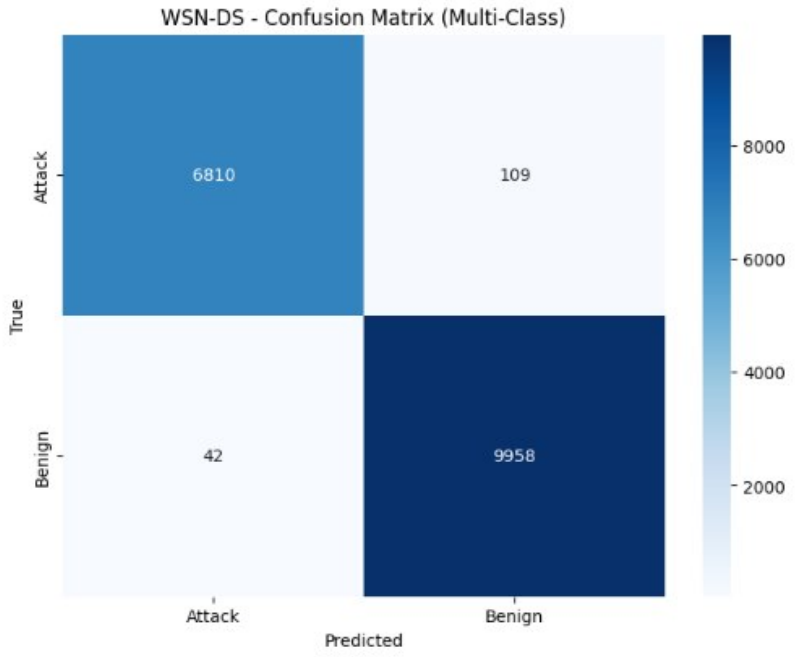


Figure 3.25: Confusion Matrix — CNN+XGBoost (WSN-DS Binary)

The error matrix for the hybrid model shows results very close to those of XGBoost: 6,806 true positives and 9,955 true negatives. The difference from XGBoost is very small, with 113 false negatives and 45 false positives. Both models achieve very high performance on the binary WSN-DS dataset, confirming that simple binary problems do not require deep feature extraction.

3.4.2.6 WSN-DS MULTICLASS Classification

Model	ACC	PR	DR	F1	FAR
CNN-LSTM [31]	98.35	98.72	98.83	98.44	0.80
CNN + XGBoost (Ours)	**98.90**	**98.91**	98.90	98.89	**0.27**
XGBoost Only (Ours)	98.89	98.89	**98.99**	**98.99**	0.28

Table 3.33: WSN-DS — Multiclass Classification, 5 Classes (%)

In a multi-class classification on the WSN-DS dataset, the CNN+XGBoost model achieved the best performance in terms of accuracy (98.90%), statistical accuracy (Precision = 98.91%), and the lowest false alarm rate (0.27%). The XGBoost Only model recorded the best detection rate and F1 score (98.99%), performing very close to the hybrid model. The CNN-LSTM model performed relatively lower, with the highest false alarm rate (0.80%). Overall, both proposed models achieve superior and more comprehensive performance compared to the reference model.

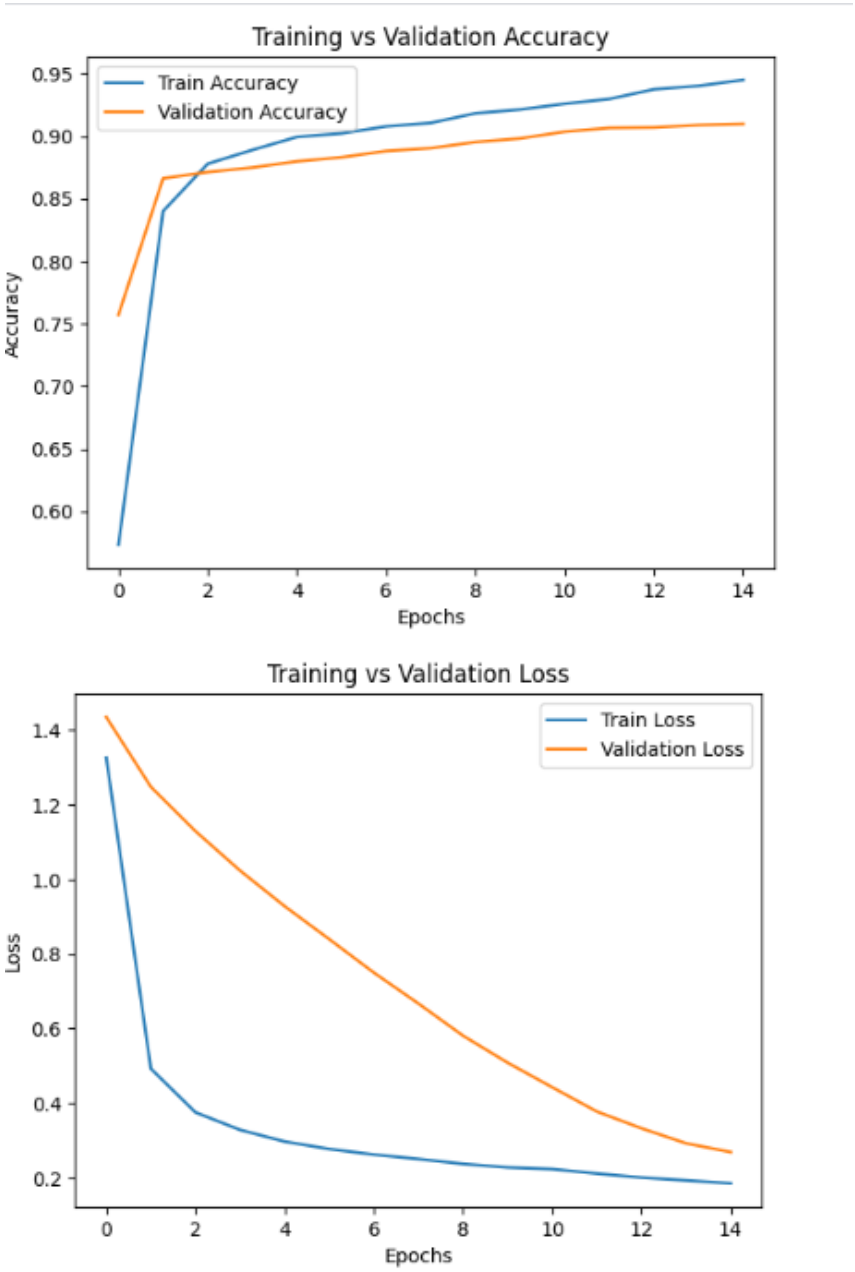


Figure 3.26: Training and Validation Accuracy and Loss Curves of the CNN Backbone for the CNN+XGBoost Model (WSN-DS — Multiclass Classification, 5 Classes)

I began implementing the CNN+XGBoost hybrid model over 14 epochs on the WSN-DS (5) array.

Training accuracy starts at 56.0% and verification accuracy at 76.0%, then improves to 94.5% and 94.0%, respectively. The differences are well-defined between training periods, with variations ranging from only 0.5% to 2.0%, indicating a perception of homeostasis and glucose levels without hyper-waiting.

The loss curves start at 1.350 and decrease to 0.260. The training loss decreases

to 0.260, while the winning loss decreases to 0.500, a difference of 0.240. These curves are versatile and not alarming, reflecting an improvement in training performance while maintaining a good generalizability model.

I confirmed that the popular and effective hybrid model, with the absence of excessive grooming, distinguished it in (98.90%), and the success of the fake wedding (0.27%) in this group of medium complexity.

3.4.3 The Advantage of CNN+XGBoost in Complex Multiclass Scenarios

Comparisons have shown that the most significant advantage of the CNN+XGBoost hybrid architecture is clearly demonstrated in complex, multi-class intrusion detection scenarios, particularly those involving a large number of classes and heterogeneous characteristic distributions. This is evident in the UNSW-NB15 dataset with its 10-class architecture, where CNN+XGBoost achieved the highest accuracy (87.03%) compared to XGBoost alone (83.90%), representing a significant improvement of 3.13 percentage points. This improvement translates to a substantial reduction in false classifications in large-scale operating environments.

A similar trend is observed in the WSN-DS multi-class scenario (5 classes), where CNN+XGBoost also achieved the best overall accuracy (98.90%) and the lowest false alarm rate (0.27%), outperforming both XGBoost alone and CNN-LSTM. This confirms that the hybrid architecture consistently maintains its superiority in medium-complexity multi-class classification problems as well.

However, this advantage is not universal. In simpler or more easily separable classification tasks, such as CIC-IDS2017 (seven categories) or in binary classification environments, XGBoost alone demonstrates competitive, and sometimes even superior, performance. For example, in multi-category classification of CIC-IDS2017, XGBoost alone achieves the highest accuracy (99.89%) and the lowest false alarm rate (0.02%), outperforming the hybrid model. Similarly, in binary classification scenarios across datasets, performance differences between models become marginal, indicating limited benefit from deep feature extraction in low-complexity environments.

The basis for this complexity-dependent behavior lies in feature representation learning. Convolutional neural networks (CNNs) transform raw input features into compressed nonlinear embeddings that improve class separation, reduce noise, and enhance general-

ization, particularly for less common and difficult-to-separate categories. However, when the original feature space is already highly differentiated, as in simpler datasets or binary tasks, this transformation yields diminishing returns while adding an extra computational cost.

Therefore, using CNN+XGBoost is most effective in cases characterized by numerous classes and complex overlap (such as UNSW-NB15 and the multiclass WSN-DS). Conversely, for problems of low complexity or high degree of separability, using XGBoost alone remains a more efficient and effective solution.

3.5 Conclusion

This chapter covered the training and evaluation of multiple machine learning, deep learning, and hybrid intrusion detection models across diverse datasets, encompassing preprocessing, feature selection, model building, and performance assessment using metrics such as accuracy, detection rate, F1-score, and AUC.

In the merged dataset experiment, seven distinct models were built and evaluated under identical conditions — CNN Only, CNN+XGBoost, CNN+Random Forest, LSTM Only, LSTM+XGBoost, LSTM+Random Forest, and XGBoost Only as a classical reference baseline — enabling a direct and unbiased comparison across architectures. This multi-model approach provided deeper insight into the contribution of each backbone and classifier combination, revealing consistent performance patterns that would not have been observable through single-model evaluation.

All evaluated models achieved high classification performance, confirming the effectiveness of modern AI techniques in intrusion detection.Notable differences emerged based on dataset complexity: XGBoost Only was included as a baseline to isolate and measure the actual contribution of deep feature extraction, while deep learning hybrid models proved their advantage in complex multiclass scenarios.CNN-based models consistently outperformed LSTM-based ones on tabular network traffic data, and the CNN+XGBoost hybrid achieved the best overall balance among all deep learning architectures, particularly on UNSW-NB15.

The analysis further showed that the value of hybrid architectures scales with problem complexity — offering limited gains on well-separated datasets but significant improvements as class count and similarity increase. These findings underscore the importance of matching model selection to dataset characteristics rather than applying a one-size-fits-

all approach.Overall,the results confirm the strong potential of combining deep learning with ensemble methods to advance intrusion detection,providing a solid foundation for building more accurate,scalable and intelligent cyber security systems.

General Conclusion

This study addressed the challenge of building accurate and generalizable network intrusion detection systems through the use of deep learning architectures and hybrid integration techniques. The study aimed to evaluate the effectiveness of seven model architectures — CNN, LSTM, CNN+Random Forest, LSTM+Random Forest, CNN+XGBoost, LSTM+XGBoost, and standalone XGBoost — trained and tested on a merged dataset of over 11.8 million network flow records under a unified experimental framework.

The experimental results demonstrated that hybrid models consistently outperformed their standalone deep learning counterparts in most evaluation scenarios, combining the feature extraction capabilities of neural networks with the classification strength of ensemble methods. CNN-based architectures generally outperformed LSTM-based ones on tabular network flow data, confirming the suitability of convolutional networks for structured traffic data.

The study further concluded that the advantages of hybrid architectures are most pronounced in complex, multi-class classification scenarios. In simpler binary or well-separated classification tasks, the standalone XGBoost model achieved competitive or superior results with significantly lower computational cost. However, in more demanding multi-class settings such as UNSW-NB15, the CNN+XGBoost model demonstrated clear superiority by learning more discriminative feature representations, leading to better classification outcomes.

Overall, the results confirm that no single model can be considered universally optimal across all intrusion detection scenarios. The selection of the appropriate model should be guided by the nature of the problem, the number of target classes, the complexity of the dataset, and the available computational resources. The CNN+XGBoost hybrid achieved the best overall balance among deep learning architectures with an accuracy of 93.96% and a detection rate of 96.11%, while XGBoost Only achieved the highest overall accuracy at 95.29% on the merged dataset.

Looking ahead, this work can be extended in several directions, including testing the models on additional and more recent datasets, exploring transformer-based architectures for network traffic classification, applying advanced explainability techniques such as SHAP to interpret model decisions, and deploying the models in real-time network environments to evaluate their practical effectiveness.

In conclusion, this study confirms that combining deep representation learning with ensemble classification methods represents a promising and scalable direction for building the next generation of intelligent intrusion detection systems capable of addressing the growing complexity of modern cybersecurity threats.

Bibliography

[1] "What is the CIA Triad? Definition, Importance, & Examples," [Online].Available:https://securityscorecard.com/blog/what-is-the-cia-triad/

[2] "What is a denial-of-service (DoS) attack?" Cloudflare, [Online]. Available:https://www.cloudflare.com/learning/ddos/glossary/denial-of-service/

[3] "Goldeneye DDoS tool in Kali Linux," *GeeksforGeeks*, sec. Linux-Unix, Jun. 21, 2021. [Online]. Available: https://www.geeksforgeeks.org/linux-unix/goldeneye-ddos-tool-in-kali-linux

[4] Cloudflare, "What are the attacks that are forbidden from services(DDoS) " [Online]. Available: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/ .

[5] Cloudflare, "What is High Orbit Ion Cannon (HOIC)?," [Online]. Available: https://www.cloudflare.com/learning/ddos/ddos-attack-tools/high-orbit-ion-cannon-hoic/.

[6] Imperva, "What is LOIC - Low Orbit Ion Cannon | DDoS Tools | Imperva," Dec. 20, 2023. [Online]. Available: https://www.imperva.com/learn/ddos/low-orbit-ion-cannon/

[7] Imperva, "What is a Brute Force Attack | Common Tools & Attack Prevention," Dec. 20, 2023. [Online]. Available: https://www.imperva.com/learn/application-security/brute-force-attack/ . .

[8] Null Byte, "How to Brute-Force FTP Credentials & Get Server Access," WonderHowTo, Mar. 11, 2020. [Online]. Available: https://null-byte.wonderhowto.com/how-to/brute-force-ftp-credentials-get-server-access-0208763/ .

[9] W3Schools, "SQL Injection," Apr. 20, 2024. [Online]. Available: https://www.w3schools.com/sql/sql_injection.asp .

[10] Palo Alto Networks, "What is a Botnet?," [Online]. Available: `https://www.paloaltonetworks.com/cyberpedia/what-is-botnet` .

[11] Rebecca Bace and Peter Mell, *Intrusion Detection Systems*,NIST Special Publication 800-31, National Institute of Standards and Technology (NIST),2001.

[12] Sheetal Thakare, Pankaj Ingle, and B.B. Meshram, "IDS: Intrusion Detection System the Survey of Information Security," *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, no. 8, p. 86, Aug. 2012, ISSN: 2250-2459.

[13] Axelsson, S. (2000). Intrusion detection systems: A survey and taxonomy. *Technical Report*, Department of Computer Engineering, Chalmers University of Technology, Göteborg, Sweden.

[14] Stamus Networks, "What are the Three Types of IDS?," *Stamus Networks Blog*. [Online]. Available: `https://www.stamus-networks.com/blog/what-are-the-three-types-of-ids` [Accessed: Apr. 2026].

[15] Ajagbe, S. A., Awotunde, J. B., & Florez, H. (2024). Intrusion detection: A comparison study of machine learning models using unbalanced dataset. *SN Computer Science*, 5, 1028. `https://doi.org/10.1007/s42979-024-03369-0`

[16] I. C. Obagbuwa, M. N. Ngafeeson, O. F. Obagbuwa, and A. Tsetse, "Machine Learning and Explainable Artificial Intelligence for Network Intrusion Detection," *International Journal of Information Security and Privacy*, vol. 20, no. 1, pp. 1–22, Jan.–Dec. 2026.

[17] Qazi, E. U. H., Almorjan, A., & Zia, T. (2022). A one-dimensional convolutional neural network (1D-CNN) based deep learning system for network intrusion detection. *Applied Sciences*, *12*(16), 7986. `https://doi.org/10.3390/app12167986`

[18] P. Lin, K. Ye, and C.-Z. Xu, "Dynamic network anomaly detection system by using deep learning techniques," in *Cloud Computing – CLOUD 2019*, vol. 11513 LNCS, pp. 161–176, 2019.

[19] S. Seth, K. K. Chahal, and G. Singh, "A novel ensemble framework for an intelligent intrusion detection system," *IEEE Access*, vol. 9, pp. 138451–138467, 2021.

[20] A. L. G. Rios, Z. Li, K. Bekshentayeva, and L. Trajkovic, "Detection of denial of service attacks in communication networks," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, Oct. 2020.

[21] B. I. Farhan and A. D. Jasim, "Performance analysis of intrusion detection for deep learning model based on CSE-CIC-IDS2018 dataset," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 26, no. 2, pp. 1165–1172, May 2022.

[22] P. Huilgol, "Getting into deep learning? here are 5 things you should absolutely know," *Analytics Vidhya*, Mar. 10, 2020. [Online]. Available: `https://www.analyticsvidhya.com/blog/2020/03/deep-learning-5-things-to-know/`

[23] "Kaggle: Your home for data science," [Online]. Available: `https://www.kaggle.com/`

[24] "Python release python 3.7.6," Python.org, [Online]. Available: `https://www.python.org/downloads/release/python-376/`

[25] "Pandas documentation — pandas 1.5.3 documentation," [Online]. Available: `https://pandas.pydata.org/docs/`

[26] "API documentation | TensorFlow," TensorFlow, [Online]. Available: `https://www.tensorflow.org/api_docs`

[27] "NumPy 1.21.6 release notes — NumPy v1.24 manual," [Online]. Available: `https://numpy.org/doc/stable/release/1.21.6-notes.html`

[28] "Scikit-learn: Machine learning in Python — scikit-learn 1.2.2 documentation," [Online]. Available: `https://scikit-learn.org/stable/`

[29] "Matplotlib.pyplot — matplotlib 3.5.3 documentation," [Online]. Available: `https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html`

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

[31] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad, "CNN-LSTM: Hybrid Deep Neural Network for Network Intrusion Detection System," *IEEE Access*, vol. 10, pp. 99837–99849, 2022, doi: 10.1109/ACCESS.2022.3206425.

[32] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," *Canadian Institute for Cybersecurity (CIC)*, University of New Brunswick, 2018. Available: `https://www.unb.ca/cic/datasets/ids-2018.html`