T.C.

GÜMÜŞHANE ÜNİVERSİTESİ

GRADUATE EDUCATION INSTITUTE

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND INTELLIGENT SYSTEMS

PERFORMANCE EVALUATION OF DEEP LEARNING MODELS FOR INTRUSION DETECTION USING NETWORK TRAFFIC

MASTER'S THESIS

Rachid CHEİCK MOHAMED

JULY-2026

GÜMÜŞHANE

T.C.

GÜMÜŞHANE ÜNİVERSİTESİ

GRADUATE EDUCATION INSTITUTE

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND INTELLIGENT SYSTEMS

AĞ TRAFIĞI KULLANARAK SALDIRI TESPİT EDEBİLEN DERIN ÖĞRENME MODELLERİNİN PERFORMANS DEĞERLENDIRMESİ

PERFORMANCE EVALUATION OF DEEP LEARNING MODELS FOR INTRUSION DETECTION USING NETWORK TRAFFIC

MASTER'S THESIS

Rachid CHEİCK MOHAMED

JULY-2026

GÜMÜŞHANE

GÜMÜŞHANE ÜNİVERSİTESİ
Lisansüstü Eğitim Enstitüsü

T.C.

GÜMÜŞHANE ÜNİVERSİTESİ

GRADUATE EDUCATION INSTITUTE

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND INTELLIGENT SYSTEMS

AĞ TRAFIĞI KULLANARAK SALDIRI TESPİT EDEBİLEN DERIN ÖĞRENME MODELLERİNİN PERFORMANS DEĞERLENDİRMESİ

PERFORMANCE EVALUATION OF DEEP LEARNING MODELS FOR INTRUSION DETECTION USING NETWORK TRAFFIC

MASTER'S THESIS

Rachid CHEICK MOHAMED

Supervisor: Asst. Prof. Dr. Samet TONYALI

JULY-2026

GÜMÜŞHANE

APPROVAL PAGE

This study, entitled "**Performance Evaluation of Deep Learning Models for Attack Detection Using Network Traffic**," prepared by **Rachid CHEICK MOHAMED** under the supervision of **Dr. Samet TONYALI**, was **unanimously** deemed successful following the master's thesis defense examination held on **03/07/2026**, and was accepted by our jury as a Master's thesis.

……...
Asst. Prof. Dr. Ramazan Özgür DOĞAN (Chair)

……...
Asst. Prof. Dr. Samet TONYALI (Supervisor)

……...
Asst. Prof. Dr. Gökhan ÇETİN (Member)

The bound copy of this thesis, which was accepted upon successful defense in the graduate thesis examination, was reviewed at the Institute Executive Board meeting dated ...../...../..... and numbered ..... / ....., and was approved as being in compliance with the thesis writing guide.

Prof. Dr. Duygu ÖZDEŞ
Director of the Institute

DECLARATION OF SCIENTIFIC ETHICS

I hereby declare that this thesis, titled **"Performance Evaluation of Deep Learning Models Capable of Detecting Attacks Using Network Traffic"**, which I have prepared as a Master's Thesis, is entirely my own work, that I have cited every source for all quotations, that I have listed all referenced works in the bibliography, and that I have reported it in accordance with the plagiarism detection software licensed by Gümüşhane University and the criteria established by the Graduate Education Institute. I confirm that I grant permission for both the paper and electronic copies of my thesis to be stored in the archives of the Gümüşhane University Graduate Education Institute. I respectfully request that the necessary actions be taken in accordance with the relevant articles of the Graduate Education and Training Regulations.

03/07/2026

Rachid CHEICK MOHAMED

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my supervisor, **Asst. Prof. Dr. Samet TONYALI**, for his guidance, support, and valuable feedback throughout the preparation of this thesis. His expertise and constructive advice have been instrumental in shaping the direction and quality of this work.

I would like to extend my deepest appreciation to **Gümüşhane University** and the **Graduate Education Institute** for providing me with the academic environment and resources necessary to carry out this research. The opportunity to pursue my Master's degree in the Department of Artificial Intelligence and Intelligent Systems has been a truly enriching experience.

I am also grateful to the faculty members and staff of the department for their support and encouragement throughout my studies.

My heartfelt thanks go to my family, whose unconditional love, patience, and sacrifices have been my greatest source of strength. Coming from Cameroon to pursue graduate studies in Turkey would not have been possible without their unwavering belief in me, and I dedicate this work to them.

I would also like to acknowledge the researchers and institutions who made the **CIC-IDS2017** and **UNSW-NB15** datasets publicly available, as this work would not have been possible without these essential resources.

Finally, I express my gratitude to **Allah** for granting me the perseverance, health, and clarity of mind to complete this journey.

Rachid CHEICK MOHAMED

GÜMÜŞHANE-2026

ABSTRACT

Digital infrastructures are growing rapidly and cyberattacks are becoming increasingly sophisticated. As a result, Network-based Intrusion Detection Systems have become a critical component of modern cybersecurity. Signature-based methods sometimes fail to detect new threats, where machine learning and deep learning can provide a valuable alternative. This study examines a one-dimensional Convolutional Neural Network and a hybrid model called CNN-BiLSTM, investigating how both architectures can detect network attacks in binary and multiclass classification settings. Both architectures were evaluated on the CIC-IDS2017 and UNSW-NB15 benchmark datasets through a rigorous experimental protocol incorporating stratified 5-fold cross-validation, Focal Loss, QuantileTransformer scaling, and SMOTE oversampling. Random Forest and XGBoost were additionally tested on UNSW-NB15 as classical baselines. The results show that CNN achieves near-perfect binary classification performance on CIC-IDS2017, with 99.75% accuracy and a ROC-AUC of 0.9999. CNN-BiLSTM reduces false negatives by approximately 36% in the binary UNSW-NB15 setting, albeit at the cost of 4–6 times higher inference latency. In multiclass classification, CNN outperforms CNN-BiLSTM on CIC-IDS2017 with a higher macro F1-Score (0.701 vs. 0.671). XGBoost achieves comparable F1 performance to deep learning models in significantly less training time, once again demonstrating the enduring value of ensemble methods. Minority attack class detection remains a fundamental unresolved challenge across all configurations, laying the groundwork for future research on class-wise threshold calibration and Transformer-based architectures. The findings provide actionable insights for practitioners choosing between deep learning and classical approaches under real-world NIDS deployment constraints.

Keywords: Network Intrusion Detection System, Convolutional Neural Network, CNN-BiLSTM, Focal Loss, Multiclass Classification.

SUMMARY

TABLES

FIGURES LIST

ABBREVIATION

ADASYN	Adaptive Synthetic Sampling
AI	Artificial Intelligence
AUC	Area Under the Curve
BiLSTM	Bidirectional Long Short-Term Memory
CNN	Convolutional Neural Network
CNN1D	One-Dimensional Convolutional Neural Network
CV	Cross-Validation
DDoS	Distributed Denial of Service
DL	Deep Learning
DNN	Deep Neural Network
DoS	Denial of Service
DR	Detection Rate
FC	Fully Connected
FFNN	Feed Forward Neural Network
FL	Focal Loss
FPR	False Positive Rate
FTP	File Transfer Protocol
GAP	Global Average Pooling
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HIDS	Host-based Intrusion Detection System
IDS	Intrusion Detection System
IIoT	Industrial Internet of Things
IoT	Internet of Things
KDD	Knowledge Discovery in Databases
KNN	K-Nearest Neighbors
LSTM	Long Short-Term Memory
MI	Mutual Information
ML	Machine Learning
MLP	Multi-Layer Perceptron
NIDS	Network Intrusion Detection System
NSL-KDD	Network Security Laboratory – Knowledge Discovery in Databases
PCA	Principal Component Analysis
PII	Personally Identifiable Information
PR-AUC	Precision-Recall Area Under the Curve
ReLU	Rectified Linear Unit
RF	Random Forest
RL	Reinforcement Learning

RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
ROC-AUC	Receiver Operating Characteristic – Area Under the Curve
R2L	Remote to Local
SA-BO-CNN	Sparse Autoencoder – Bayesian Optimization – Convolutional Neural Network
SMOTE	Synthetic Minority Over-sampling Technique
SOC	Security Operations Center
SQL	Structured Query Language
SSH	Secure Shell
SVM	Support Vector Machine
TPR	True Positive Rate
U2R	User to Root
UNSW-NB15	University of New South Wales Network-Based 2015
XGBoost	Extreme Gradient Boosting
XSS	Cross-Site Scripting

Chapter 1

1 Introduction

The rapid expansion of digital infrastructures and the exponential growth of interconnected systems have fundamentally transformed the landscape of cybersecurity threats. Modern networks handle billions of data packets daily, spanning critical sectors such as finance, healthcare, energy, and public administration. This massive scale creates an environment where malicious actors continuously develop and refine sophisticated attack strategies, exploiting vulnerabilities in communication protocols, application layers, and human behavior alike. Against this backdrop, the ability to detect intrusions accurately, rapidly, and reliably has become one of the central challenges of applied computer science.

This thesis addresses this challenge by investigating the application of deep learning architectures to the problem of Network Intrusion Detection Systems (NIDS). Specifically, it evaluates and compares two neural network models, a one-dimensional Convolutional Neural Network (CNN1D) and a hybrid CNN-BiLSTM architecture, across two widely used benchmark datasets: CIC-IDS2017 and UNSW-NB15. The evaluation spans both binary and multiclass classification tasks, complemented by classical machine learning baselines (Random Forest and XGBoost) to contextualize the results within the broader NIDS literature.

1.1 Problem description and context

Intrusion detection refers to the process of monitoring network traffic and system activity to identify events that may indicate a security breach, policy violation, or malicious attempt to compromise system integrity. Traditional approaches to intrusion detection relied heavily on rule-based or signature-based systems, which match observed traffic patterns against known attack signatures stored in a database. While effective against known threats, these methods are fundamentally reactive: they fail to detect novel attack variants, zero-day exploits, and behavioral anomalies that fall outside their predefined rule sets.

The limitations of signature-based methods have driven significant research interest toward machine learning (ML) and deep learning (DL) approaches, which can learn discriminative features directly from traffic data without relying on hand-crafted rules. The ability of neural networks to automatically extract hierarchical representations from high-

dimensional, heterogeneous network flow data makes them particularly promising candidates for this application.

1.2 The importance of cybersecurity and NIDS in IDS

According to the (Magazine, 2018) cybersecurity has become a strategic priority for organizations during the digital era. Global cybercrime is expected to cost over 8 trillion USD by 2023, increasing to more than 10 trillion USD by 2025. Cyberattacks can impact not only financial information but also affect critical infrastructure, result in the personal information of millions of individuals being compromised or stolen, and destroy trust in digital services. Well-known cyberattacks include the SolarWinds supply chain attack reported by (Coco et al., 2022), the Colonial Pipeline ransomware attack published in (*Colonial Pipeline*, s. d.), as well as numerous cybersecurity breaches and attacks targeting the healthcare and government sectors. Within this context, Network Intrusion Detection Systems occupy a central position in any layered cybersecurity defense strategy. Deployed at strategic network vantage points, NIDS passively monitor traffic and alert security operations teams to suspicious activity. Unlike preventive controls such as firewalls, which enforce access policies at network boundaries, NIDS provide continuous visibility into traffic behavior, enabling the detection of threats that have already bypassed perimeter defenses. Their effectiveness directly conditions the speed and quality of incident response.

The transition from traditional to AI-powered NIDS is no longer a research horizon but an operational necessity. The volume and velocity of modern network traffic make manual analysis impossible, and the sophistication of advanced persistent threats (APTs) renders static rule-based systems insufficient. Machine learning-based NIDS that can adapt, generalize, and scale represent the next generation of network security monitoring tools, and their rigorous evaluation constitutes a meaningful contribution to both the research community and the cybersecurity industry.

1.3 Motivations

The motivations behind this research arise from two converging observations: one academic, and one technical.

From an academic standpoint, the existing NIDS literature, while rich, suffers from a lack of systematic cross-architecture comparisons under controlled and reproducible experimental conditions. Many published results are difficult to compare due to inconsistent preprocessing pipelines, varying train/test splits, and different evaluation metrics. This thesis

aims to contribute a rigorous, methodologically coherent comparative evaluation that can serve as a reliable reference point for future work.

From a technical standpoint, the BiLSTM architecture, which processes sequential input in both forward and backward directions, offers a theoretically appealing capability for network traffic analysis: the ability to capture temporal dependencies at the flow level in a bidirectional manner. Yet its practical advantage over simpler convolutional architectures for tabular network flow data remains empirically underexplored. The present work provides a direct, controlled comparison that quantifies this advantage across tasks and datasets.

1.4 Problem statement

The central problem addressed in this thesis can be stated as follows: despite the proliferation of deep learning-based NIDS proposals in the literature, it remains unclear which architectural families offer the best trade-off between detection performance, false positive rate, attack-type discrimination, and inference speed across different benchmark conditions. Specifically, three unresolved tensions motivate this investigation.

The first tension concerns model complexity versus generalization. More expressive architectures such as CNN-BiLSTM carry greater representational capacity, but this advantage may not translate into superior performance on tabular network flow data where temporal dependencies at the sample level are limited. Understanding when added complexity yields genuine gains is essential for practical model selection.

The second tension concerns global accuracy versus class-level discrimination. High overall accuracy in the presence of class imbalance can mask poor detection of minority attack classes, which may be precisely those of greatest operational concern. Optimizing for accuracy alone is therefore insufficient, and a rigorous evaluation must account for the performance distribution across all traffic categories.

The third tension concerns the role of classical baselines. The deep learning community has sometimes treated classical ML models as superseded by neural approaches, yet for datasets of moderate size with well-engineered features, Random Forest and XGBoost remain highly competitive. The conditions under which deep learning provides a clear and measurable advantage over these baselines in the NIDS domain deserve explicit investigation.

1.5 Research question

In light of the problem formulation above, this thesis is guided by the following research questions:

RQ1: To what extent do CNN and CNN-BiLSTM architectures achieve high accuracy, low false positive rate, and strong attack-type discrimination on the CIC-IDS2017 and UNSW-NB15 benchmark datasets, in both binary and multiclass classification settings?

RQ2: Does the hybrid CNN-BiLSTM architecture provide a statistically meaningful and operationally relevant performance advantage over the simpler CNN architecture, and under what dataset or task conditions is this advantage most pronounced?

RQ3: How do deep learning-based NIDS models compare against well-optimized classical machine learning baselines (Random Forest, XGBoost) in terms of detection performance and computational efficiency on the same benchmark dataset?

1.6 Organization

The remainder of this thesis is organized as follows. Table 1 summarizes the overall structure of our present work, providing an overview of each chapter and its content.

Table 1: Chapter and content.

Chapter	Content
Chapter 1	**Introduction**. Presents the research context, problem statement, motivations, research questions, and the overall structure of the thesis.
Chapter 2	**Theoretical Background**. Provides a comprehensive review of machine learning fundamentals, neural network architectures, and the principal concepts underlying intrusion detection systems. Covers supervised learning, feature engineering, class imbalance handling, and evaluation metrics.
Chapter 3	**Deep Learning Architectures for NIDS**. Describes in detail the CNN and CNN-BiLSTM architectures investigated in this thesis, including their design principles, convolutional and recurrent components, and the rationale for their application to network traffic data.
Chapter 4	**Experimental Results and Analysis**. Presents the experimental setup, datasets, preprocessing pipeline, and a comprehensive evaluation of all model configurations. Results are reported and interpreted for each architecture, dataset, and classification task, with a dedicated subsection for the classical ML baselines.
Chapter 5	**Conclusion and Future Work**. Synthesizes the main findings, addresses the research questions, acknowledges the limitations of the study, and outlines directions for future research including threshold optimization, architecture extensions, and real-world deployment considerations.

1.7 Ethics and sustainability

The research conducted in this thesis raises no direct ethical concerns with respect to data collection or human subjects, as all experiments are performed exclusively on publicly available, pre-recorded network traffic datasets (CIC-IDS2017 and UNSW-NB15). These datasets do not contain personally identifiable information (PII) and were released by their respective institutions for the explicit purpose of academic research in the cybersecurity domain.

However, a broader set of ethical and sustainability considerations deserves explicit acknowledgement. With respect to algorithmic fairness, the evaluation metrics adopted in this thesis, particularly the macro-averaged F1-Score and per-class recall, are specifically designed to expose performance disparities across traffic categories. This choice reflects a deliberate commitment to transparency: a model that achieves high average accuracy by systematically failing on rare but operationally critical attack classes is not considered acceptable, regardless of its aggregate performance.

Chapter 2

2 Background

2.1 Introduction

With the rapid proliferation of connected systems and the exponential growth of data flowing across networks, cyberattacks targeting network infrastructure have become one of the most pressing threats faced by organizations today. In response to this reality, the research community has developed a wide variety of intrusion detection systems (IDS), which have progressively evolved from signature-based approaches toward more adaptive methods grounded in machine learning (ML) and, more recently, deep learning (DL). This chapter provides a structured review of existing approaches, the most widely used datasets in the literature, the limitations identified across prior work, and the broader context motivating the present research.

.

2.2 Network Traffic Analysis and Intrusion Detection Systems

2.2.1 Definition and role of an Intrusion Detection Systems (IDSs)

Intrusion Detection is a security service that monitors and analyzes system events to find, and provide real-time or near-real-time warnings of attempts to access system resources in an unauthorized manner (Farhan et al., 2025a). In other words, an Intrusion Detection System (IDS) is generally defined as a security mechanism responsible for monitoring, analyzing, and identifying suspicious or malicious activities within a computer system or network. Its primary objective is to detect abnormal behavior, security policy violations, or attack attempts that could compromise the confidentiality, integrity, or availability of resources.

An Intrusion Detection System (IDS) works by examining network traffic, system logs, or host behavior in real time or retrospectively, and then comparing this information to known attack signatures or statistical/algorithmic models that detect anomalies. Modern IDSs increasingly use advanced techniques such as machine learning and deep learning to improve accuracy, reduce false positives, and identify new or complex attacks.

2.2.2 Types of IDS

2.2.2.1 Network-based Intrusion Detection System

A Network Intrusion Detection System (NIDS) is a system designed to analyze network traffic in real time. The NIDS examines traffic packet by packet in real time, or close to real time, to attempt to detect intrusion patterns. The NIDS may examine network-, transport-, and/or application-level protocol activity (Brown, 2008). A key feature of the NIDS is its ability to monitor multiple hosts simultaneously, making it even more effective compared to other systems.

2.2.2.2 Host-based Intrusion Detection System

A Host-based Intrusion Detection System (HIDS) is installed directly on a host (server, computer, IoT device). It analyzes internal system activities, such as log files, processes, file access, login attempts, and system modifications. It essentially acts as an agent software directly embedded in a device. This system performs in-depth analysis of several things, including log files, running processes, and the integrity of system files.

2.2.2.3 Hybrid IDS

A Hybrid IDS combines the features of a NIDS and a HIDS, providing a comprehensive view of security. It correlates data from both network traffic and internal host events, improving detection accuracy and reducing false positives. It can fuse data from multiple sources simultaneously, mitigating the individual limitations of NIDS and HIDS.

2.2.3 Detection methods

2.2.3.1 Anomaly-based

Anomaly detection is the process of identifying data points that significantly deviate from the majority of samples in a dataset. Such data points are known as anomalies, and their identification and isolation can comprehensively enhance the stability and robustness of applications (Ioannidou, 2021, p. 14). In reality, IDS operates according to two principles based on this method. It learns the normal behavior of the network, and subsequently, any behavior that deviates from this profile is considered suspicious. However, it often gets lost in the detection of false positives, although it is capable of detecting zero-day attacks.

2.2.3.2 Signature-based

Signature-based detection relies on comparing network traffic or system events against a database of known attack patterns. Much like antivirus software, this approach searches for exact matches to previously documented malicious activity, making each signature unique to a particular threat. The concrete form of a signature depends largely on where the detection system is deployed. In a network-based IDS, signatures typically describe observable

characteristics at the packet level. In a host-based IDS, on the other hand, signatures operate at the system level, where they may take the form of a cryptographic hash of a critical file that has been tampered with by a rootkit, or a recognizable sequence of anomalous system calls captured in an audit.

2.3 Machine Learning approach for Intrusion Detection System

The simplest way to describe machine learning is this: instead of writing explicit rules for a computer to follow, you give it data and let it figure out the rules on its own. This idea, while conceptually straightforward, has profound implications. A machine learning model, on the other hand, learns from examples. It finds patterns, builds internal representations, and uses those representations to make predictions on new, unseen data.

This distinction becomes especially important in domains like cybersecurity. Writing a rule-based system to detect every possible type of network intrusion is practically infeasible attacks evolve, new variants appear daily, and attackers specifically design their techniques to bypass known signatures. Machine learning offers an alternative: train a model on examples of normal and malicious traffic, and let it learn to distinguish between the two. This is, at its core, the motivation behind network intrusion detection systems (NIDS) based on ML.

Formally, a machine learning algorithm can be described as a function $f: X \rightarrow Y$ that maps an input space X (the features, such as packet size, protocol type, duration) to an output space Y (the label, such as Normal or Attack). The learning process consists of finding the optimal parameters of f by minimizing a loss function over a training dataset.

Machine learning is generally divided into three main paradigms depending on the type of feedback available during training: supervised learning, unsupervised learning, and reinforcement learning. Each paradigm is suited to different types of problems, and understanding the differences between them is essential to making the right architectural choices.

Supervised learning: Supervised learning is probably the most widely used paradigm in practice, underpins most of the work in this thesis. The core idea is simple: the training data comes with labels. Each example is a pair (x, y) where x is the input (a vector of features) and y is the expected output (the correct class or value). The model is trained to minimize the discrepancy between its predictions and the true labels, and over time it learns to generalize to examples it has never seen. In the context of NIDS, supervised learning is used to train a

classifier that takes as input a feature vector describing a network flow and outputs a label indicating whether that flow is normal or corresponds to a specific attack type. In their publish paper, (Moustafa & Slay, 2016; Sharafaldin et al., 2018) use datasets like UNSW-NB15 and CIC-IDS2017 which is specifically designed for this purpose, providing labeled network traffic with both benign and attack samples across multiple attack categories.

Unsupervised learning: Unsupervised learning takes a very different approach: there are no labels. The algorithm receives only the input data X and must discover structure on its own. Rather than learning a mapping from inputs to known outputs, the model tries to understand the underlying distribution of the data finding clusters, reducing dimensions, or learning compact representations. In cybersecurity, unsupervised methods are particularly valuable for anomaly detection. Rather than classifying traffic against known attack signatures, an anomaly detector learns what normal traffic looks like and flags anything that deviates significantly from that model. This has the significant advantage of being able to detect zero-day attacks; attacks that have never been seen before and therefore cannot appear in any labeled training set. Common unsupervised techniques used in NIDS research include K-Means clustering, autoencoders (which learn a compressed representation of normal traffic and flag high reconstruction error as anomalous), and DBSCAN for density-based clustering of network flows.

Reinforcement learning: Reinforcement learning (RL) is the third paradigm, and it operates on a fundamentally different principle. An agent interacts with an environment, takes actions, and receives rewards or penalties depending on the outcome of those actions. The goal is to learn a policy, which is a mapping from states to actions that maximizes the cumulative reward over time.

In this chapter, we present two of machine learning algorithms that form the baseline our proposed NIDS: **Random Forest and XGBoost.**

Table 2: This table shown us a brief comparison of the three main machine learning paradigms and their applications in NIDS.

Paradigm	Labels	Goal	NIDS Application
Supervised	Yes	Learn input-output mapping	Attack classification (binary / multiclass)
Unsupervised	No	Discover hidden structure	Anomaly detection, zero-day threats
Reinforcement	Reward signal	Learn optimal policy	Adaptive IDS, automated response

Several classical machine learning algorithms have been widely evaluated in the context of network intrusion detection. Decision Tree have a strengths power in supervised learning fast and interpretable. The limitation of overfitting is appreciated easily with a good baseline on KDD. As for Random Forest, it's robust handles noise uses ensemble and less interpretable with 97%+ on CIC-IDS-2017. However, XGBoost uses ensemble boosting fast regularized and very strong on multiclass NIDS baseline.

Several classical machine learning algorithms have been widely evaluated in the context of network intrusion detection. Decision Trees are known for being fast and interpretable, which makes them a useful starting point. However, they tend to overfit easily, meaning they perform well on training data but struggle to generalize despite this, they still provide a decent baseline on the KDD dataset. Random Forest improves on this by combining multiple trees into an ensemble, making it more robust to noisy data. The trade-off is that it becomes harder to interpret. It remains one of the strongest classical methods, achieving over 97% accuracy on CIC-IDS2017. XGBoost also uses ensemble boosting and adds regularization to control overfitting, making it particularly effective for multiclass NIDS classification tasks.

2.4 Evaluation of Various Deep Learning Models for IDS

Deep learning is a sub-field of machine learning that relies on artificial neural networks composed of multiple processing layers.(Krizhevsky et al., 2012) introduced AlexNet, a deep convolutional network that achieved a remarkable performance gap over all competing classical methods at the 2012 ImageNet competition a result widely regarded as the moment that triggered the modern deep learning era. From that point on, deep learning became the go-to approach for almost any task involving complex, high-dimensional data. What distinguishes deep learning from classical machine learning is its ability to learn hierarchical representations directly from raw data. In this chapter, we present the two deep learning architectures that form the foundation of our proposed NIDS: the Convolutional Neural Network (CNN) and the Bidirectional Long Short-Term Memory network (BiLSTM). We conclude by explaining how we combine CNN and BiLSTM into a single hybrid model.

2.4.1 Convolutional Neural Networks (CNN)

Convolutional Neural Networks were originally designed for image processing tasks, where the spatial structure of data for example, in network intrusion detection the input is not

an image but some feature vector representation of a network flow. For their application in this context, we easy replace 2D by 1D convolutions; a convolutional filter slides over the feature vector and finds local correlations between neighboring features. E.g., a combination of high port number, small flow duration and large packet count that can only be observed for a port scanning attack would become a filter by itself.

2.4.1.1 Architecture Components

The basic CNN for NIDS is constructed of components stacked on top of each other. The convolutional layer takes some filters (which can be learned) and applies them to the input returning feature maps showcasing certain patterns, where each of these patches with different type of pattern is detected in parallel by multiple filters. Batch Normalization separates each convolutional layer by normalizing activations across the mini-batch, which stabilizes training and accelerates convergence. For Each feature map, we have to apply Max Pooling which only takes the maximum value in a sliding window while reducing its dimension and having expense-saving cost as well as giving translation invariance property.

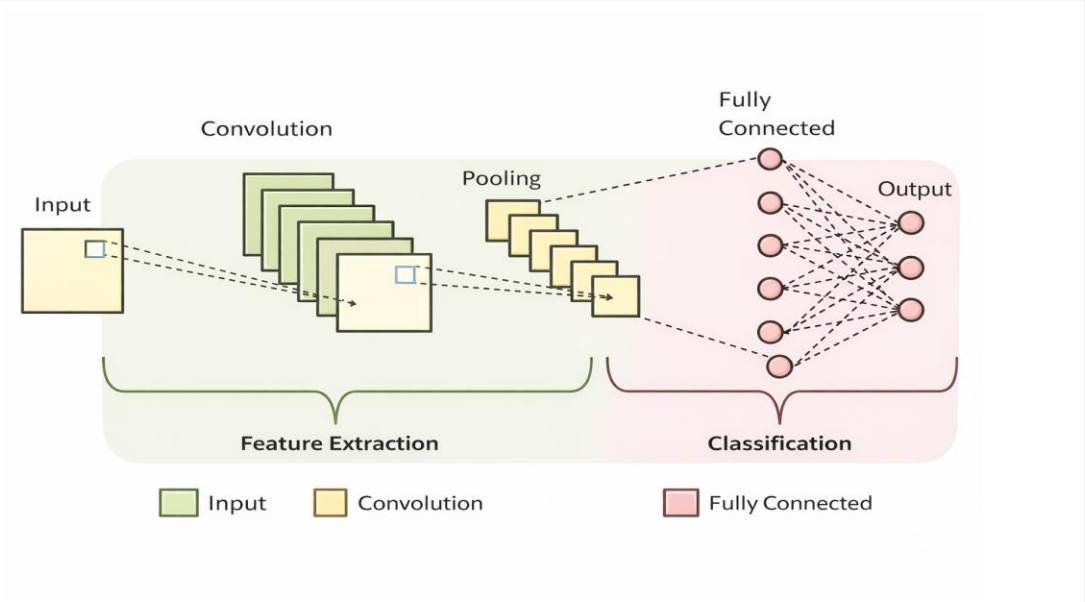


Figure 1: Convolutional neural network in the NIDS context

The main strength of CNN in the NIDS context is its ability to automatically extract relevant feature combinations from the raw traffic statistics without manual feature engineering. Its main limitation is that it does not natively model temporal dependencies between successive network flows, which is a limitation that the LSTM architecture, presented next, was specifically designed to address.

Furthermore, the intrusion detection mechanism using a CNN in an IDS. This mechanism unfolds in three main phases: automatic extraction of features, learning deep representation and traffic classification.

2.4.2 Recurrent Neural Networks (RNNs)

2.4.2.1 Motivation: The Limits of Standard RNNs

Network traffic is inherently sequential. Packets arrive in a specific order, and the behavior of a connection over time, i.e., how it starts, how it evolves, how it terminates, carries crucial information about whether it is malicious or benign. A DoS attack, for instance, is not characterized by a single suspicious packet. Instead, it is a sustained pattern of high-volume traffic that unfolds over time. To capture this kind of temporal dependency, we need a model that can maintain memory across time steps.

Recurrent Neural Networks (RNNs) were designed for this purpose. At each time step, an RNN takes as input the current observation and the hidden state from the previous step, producing a new hidden state that summarizes all past information. In principle, this allows the network to maintain a running memory of the sequence. In practice, however, standard RNNs suffer from the vanishing gradient problem: during backpropagation through time, gradients tend to shrink exponentially as they propagate through many time steps, making it very difficult to learn long-range dependencies. An RNN processing a network flow that started 50 packets ago will have essentially forgotten information about those early packets.

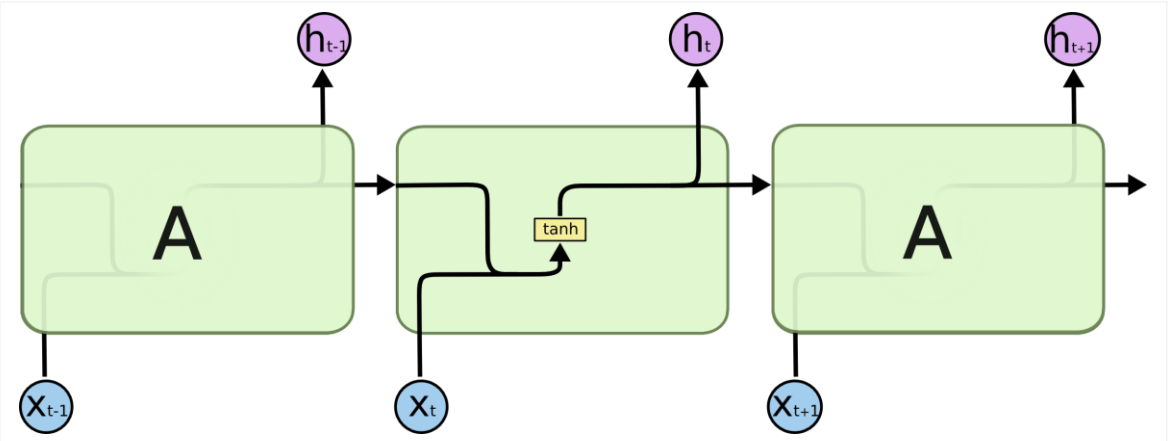


Figure 2: This figure shown us a simple representation of recurrent neural network

2.4.2.2 The LSTM Cell

In 1997, Hochreiter and Schmidhuber introduced a new architecture to the Recurrent Neural Network (RNN) and addressed the problem of vanishing gradient in regular RNNs by proposing the so-called Long Short-Term Memory network (long short-term memory network) – hence the name lntm (lstm network). In an LSTM network, every cell not only keeps a hidden

state (ht) equivalent to the RNN output, but also a cell state Ct that is updated throughout time steps via additive interactions and not multiplicative as in regular RNNs.

LSTM networks control the time constants of the internal states via the three gates $(i_t z, i, f)$ which are implemented as sigmoid-activated linear layers.

The Forget Gate decides what fraction of the previous cell state to discard. When its output is close to 0, the memory is erased; when it is close to 1, the memory is fully preserved. In network traffic terms, the forget gate might learn to reset its memory when it detects the end of one connection and the beginning of another.

The Input Gate controls how much new information from the current input is incorporated into the cell state. This is computed in two parts: a sigmoid gate that decides which values to update, and a tanh transformation that produces candidate values to potentially add.

The Output Gate determines how much of the cell state is exposed as the hidden state h_t at the current time step. This hidden state is what the network passes to the next layer or to the output.

The four core equations of a forward "normal" LSTM cell are:

$$f_t = \sigma\big(W_f.[h_{t-1}, x_t] + b_f\big) \quad \text{(Forget gate)} \quad (1)$$

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \quad \text{(Input gate)} \quad (2)$$

$$\widetilde{C_t} = tanh(W_C.[h_{t-1}, x_t] + b_c) \quad \text{(Candidate cell state)} \quad (3)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C_t} \quad \text{(Updated cell state)} \quad (4)$$

$$O_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \quad \text{(Output gate)} \quad (5)$$

$$h_t = ot \odot \tanh(C_t) \quad \text{(Hidden state output)} \quad (6)$$

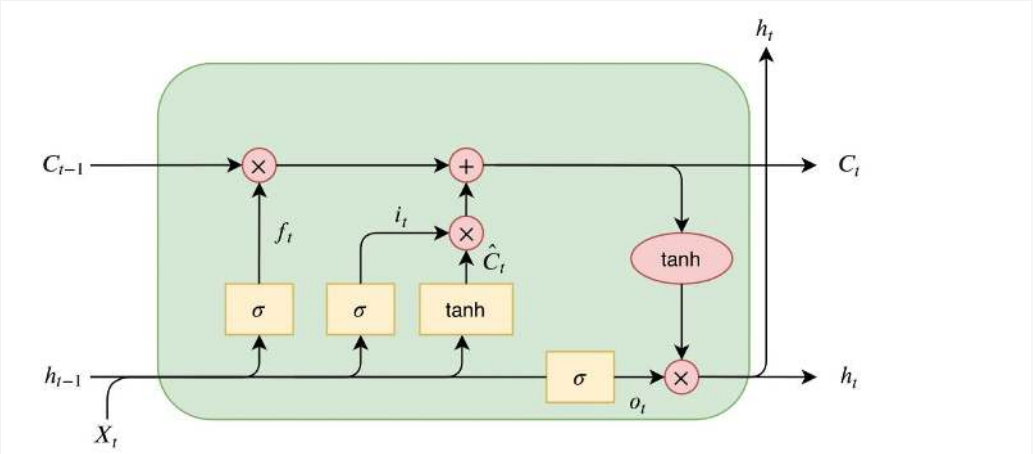


Figure 3: A An internal architecture of LSTM (Long-Short Term Memory)

This architecture gives the LSTM an exceptional ability to learn long-range dependencies. A study specifically testing BiLSTM on KDDCUP-99 and UNSW-NB15

demonstrated that LSTM-based models achieve 99% accuracy on both datasets, outperforming shallow machine learning approaches particularly on minority attack classes (Vinayakumar et al., 2019).

2.4.2.3 Bidirectional LSTM (BiLSTM)

Standard LSTM networks process sequences from first time step to last time step. Bidirectional LSTM networks process the same sequences backwards and forwards in time and for each time step both networks contribute to the output. In addition to the forward LSTM (feed forward network) there exists a backward LSTM (feedback network). The output for any time step is a combination of the two networks.

In the existing literature on recurrent neural networks, all time steps in a sequence are treated as equally important, and each element in the sequence is only defined by its prior and all preceding elements. In this paper, we explore a new direction in RNN architecture: we investigate the possibilities of the network paying attention to what follows an element in a sequence. There are many applications for this sort of architecture - natural language processing, for example, can be done by modeling words by their prior and subsequent neighbors alike. Network traffic can also be modeled this way - is a burst of packets (a spike in traffic) benign (part of a connection-teardown, for example) or malicious (potentially indicative of lateral movement), The four core equations of a backward LSTM cell are:

$f_t = \sigma\big(W_f.[h_{t+1}, x_t] + b_f\big)$ (Forget gate) (7)

$i_t = \sigma(W_i.[h_{t+1}, x_t] + b_i)$ (Input gate) (8)

$\widetilde{C_t} = tanh(W_C.[h_{t+1}, x_t] + b_c)$ (Candidate cell state) (9)

$C_t = f_t \odot C_{t+1} + i_t \odot \widetilde{C_t}$ (Updated cell state) (10)

$O_t = \sigma(W_o.[h_{t+1}, x_t] + b_o)$ (Output gate) (11)

$h_t = ot \odot \tanh(C_t)$ (Hidden state output) (12)

The forget gate $f_t = \sigma\big(W_f.[h_{t+1}, x_t] + b_f\big)$ (7) This gate decides what information to throw away from the previous cell state. The sigmoid function σ outputs a value between 0 and 1. The formula is given by: $\sigma = \frac{1}{1+e^{-x}}$. (13)

Candidate Cell State $\widetilde{C_t} = tanh(W_C.[h_{t+1}, x_t] + b_c)$ (9) computes the new candidate values that could be added to the cell state. The *tanh* function squashes values between -1 and +1, encoding the direction and magnitude of the new information. The formula is given by: $\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$. (14)

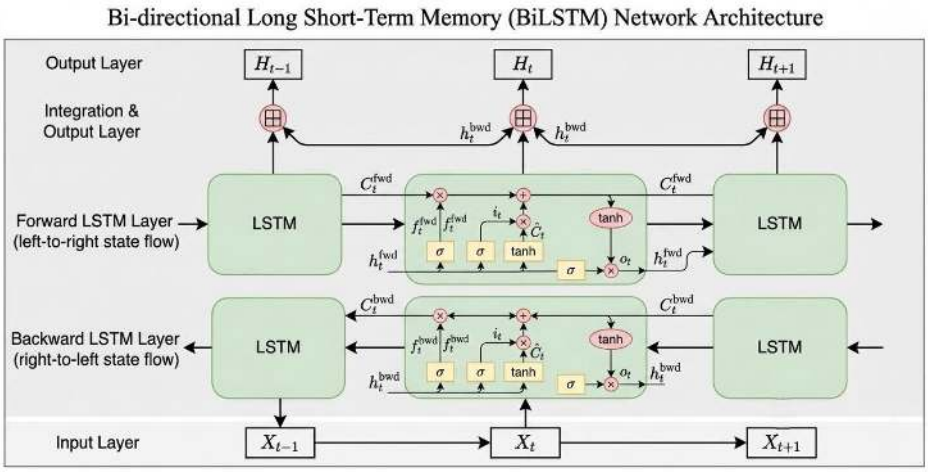


Figure 4: Bidirectional Long-Short Term Memory (BiLSTM).

2.4.2.4 BiLSTM in the NIDS Literature

The superiority of BiLSTM over standard LSTM for NIDS has been validated across multiple studies. (Udurume et al., 2024) shown us a comprehensive comparative study on NSL-KDD and UNSW-NB15 reported that a CNN-BiLSTM model achieved accuracies of 99.89% and 98.95%, respectively, outperforming traditional ML methods by a margin ranging from 1.5% to 3.5%. Another study, (Peng et al., 2023) proposed to us a CBF-IDS model combining CNN, BiLSTM, and focal loss specifically to handle class imbalance reported competitive results on NSL-KDD, UNSW-NB15, and CIC-IDS2017 simultaneously.

A particularly relevant studyfor our work combined minimal CNN with BiLSTM and chi-square cathegorical features scoring in feature selection on the UNSW-NB15 dataset, demonstrating that this combination reduces inference time while maintaining high detection accuracy a practical consideration for real-time NIDS deployment.Although they are used today, hybrid models form an essential basis for understanding deep learning-based intrusion detection.

2.4.3 Hybrid Models: CNN-LSTM

2.4.3.1 Rationale for the Hybrid Approach

Having reviewed CNN and LSTM separately, it is natural to ask whether combining them could yield better results than either alone. The answer, according to a substantial body of recent literature, is yes and the reason is that the two architectures are genuinely complementary.

CNN is excellent at detecting local patterns and spatial correlations between features within a single traffic flow. It reduces dimensionality, filters noise, and produces a compact, informative feature representation. However, it treats each flow as an isolated observation and does not model the temporal evolution of network behavior across multiple flows.

LSTM, on the other hand, excels at modeling sequential dependencies across time steps. But if fed raw, high-dimensional features directly, it may struggle to identify the most relevant patterns and can be computationally expensive.

By feeding the output of a CNN block into an LSTM block, we get the best of both worlds: the CNN extracts compact, meaningful feature representations from each flow, and the LSTM then models the temporal relationships between these representations. This hierarchical feature extraction pipeline has become one of the dominant paradigms in recent NIDS research.

2.4.3.2 Architecture of Our CNN-BiLSTM Model

Our proposed CNN-BiLSTM model consists of four stages. In the first stage, the preprocessed network flow features from UNSW-NB15 are fed as input to the CNN block. The CNN block applies two successive 1D convolutional layers: the first with 64 filters and the second with 128 filters, both with a kernel size of 3 and ReLU activation function. Each convolutional layer is followed by Batch Normalization. After the second convolutional layer, a MaxPooling operation reduces dimensionality, followed by a Dropout layer to prevent overfitting.

In the second stage, the output of the CNN block is fed into the BiLSTM block. The BiLSTM processes this compressed feature representation in both directions, capturing temporal dependencies in the network traffic sequence. The forward and backward LSTM outputs are concatenated at each time step, producing a combined representation that integrates both past and future context.

In the third stage, a fully connected Dense layer with 128 units and ReLU activation processes the BiLSTM output, followed by another Dropout layer. Finally, the output layer uses Softmax activation to produce probability distributions over the target classes either two classes (Normal/Attack) for binary classification, 10 classes for multiclass classification on UNSW-NB15 or 15 classes for CIC-IDS2017.

2.5 State of the art

2.5.1 Overview and Context

In the last decade, network intrusion detection evolved from rule-based to data-driven methods. Firstly, classical machine learning techniques such as decision trees, random forests, support vector machines (SVMs), and ensemble learning methods like AdaBoost, bagging, and stacking, have been developed and employed. Next, the interest on deep learning-based solutions such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), fully connected neural networks (FCNs), residual neural networks (Res Nets), autoencoders, generative adversarial networks (GANs), including neural network ensembles increased significantly. Recently, hybrid models consisting of machine learning and deep learning, architectures enhanced with attention mechanisms and the first attempts on graph neural networks have appeared in the literature. Works studied that are related to this thesis and discussed in detail, related to architectures, datasets, and approaches. Classical Machine Learning as a Baseline.

Before turning to deep learning approaches, it is worth acknowledging the persistent relevance of classical machine learning methods. (Türk, 2023) made a comprehensive study evaluated several machine learning algorithms on both the UNSW-NB15 and NSL-KDD datasets, achieving 98.6% and 98.3% accuracy for two-class and multiclass classification on UNSW-NB15, and 97.8% and 93.4% on NSL-KDD respectively. This work is particularly relevant because it provides a direct baseline on the same datasets used in the present thesis.

2.5.2 Single and Shallow Deep Learning Approaches

One of the most direct motivations for using deep learning is the ability to automatically extract hierarchical features from raw traffic data.

On UNSW-NB15 specifically (Farhan et al., 2025b), study from Scientific Reports, proposed a sequential Deep Neural Network (DNN) architecture using ReLU activations combined with Extra Tree Classifier-based feature selection, argued that previous DNN-based approaches suffered from vanishing gradients due to the use of sigmoid or tanh activation functions, and that ReLU combined with targeted feature selection significantly improves both interpretability and computational efficiency on this dataset. this architectural choice aligns with the general trend of the field, and the emphasis on feature selection as a preprocessing step is directly relevant to the 40-feature pipeline used in our own UNSW-NB15 experiments.

2.5.3 Feature Selection and Hybrid Deep Learning on CIC-IDS2017

Several works have focused specifically on CIC-IDS2017, often combining deep learning with explicit feature optimization. A 2023 study proposed a CNN-GRU hybrid architecture with feature selection on CIC-IDS2017, combining three CNN layers and two GRU layers with removal of redundant features. The resulting model achieved 98.73% accuracy with an FPR of 0.075. (Henry et al., 2023), this approach is noteworthy because it treats feature selection not as a preprocessing afterthought but as an integral part of the model. This is the design strategy that our own work partially addresses through quantile normalization and the inherent feature selection capacity of convolutional layers. The study also highlights that GRU, like BiLSTM, offers a solution to the long-term dependency problem that pure CNN architectures face. The difference in our work is the choice of BiLSTM over GRU. Both are valid options, but BiLSTM's, bidirectional processing provides richer context at the cost of slightly higher computational overhead.

2.5.4 Ensemble Methods and Classical Approaches Revisited

Not all recent progress has come from deep learning alone. Ensemble methods remain highly competitive, particularly when interpretability and training efficiency are priorities. A 2025 study proposed a hybrid Feed-Forward Neural Network (FFNN)-XGBoost framework for IoT intrusion detection on CIC-IoT-2023, combining principal component analysis (PCA) for dimensionality reduction with a two-step resampling strategy and achieving 99% accuracy for both binary and multiclass classification. (Alashjaee & Alqahtani, 2025), the key insight here is the complementarity between deep feature extraction (FFNN) and gradient boosting (XGBoost): The neural network learns complex representations, while XGBoost provides high-precision discrimination. This hybrid philosophy is architecturally distinct from end-to-end deep learning but achieves comparable results.

2.6 Synthesis and Positioning of the Present Work

The present work contributes to this landscape by conducting a rigorous, side-by-side comparison of CNN1D and CNN-BiLSTM under a consistent preprocessing and evaluation protocol on two benchmark datasets: CIC-IDS 2017 and UNSW-NB15, with a focus on multiclass classification, 5-fold cross-validation, and operational metrics including inference throughput and FPR. This systematic comparison, rarely done with such rigor in the reviewed literature, provides actionable insights for practitioners choosing between architectures under real-world deployment constraints.

Chapter 3

3 methodology and proposed approach

3.1 Datasets Description

This study employs two widely used benchmark datasets, **UNSW-NB15** and **CICIDS2017**, to evaluate the proposed deep learning–based intrusion detection system. These datasets are selected due to their realism, diversity of attack scenarios, and extensive use in intrusion detection research.

3.1.1 UNSW-NB15 Dataset

The UNSW-NB15 dataset was developed by the Australian Centre for Cyber Security using the IXIA PerfectStorm tool to generate realistic network traffic. It includes both normal and malicious activities, covering multiple modern attack categories such as DoS, Exploits, Fuzzers, Reconnaissance, and Backdoors. The dataset contains flow-based records described by numerical and categorical features and supports both binary and multiclass intrusion detection tasks.

3.1.2 CICIDS2017 Dataset

The CICIDS2017 dataset was created by the Canadian Institute for Cybersecurity to address the limitations of older intrusion detection datasets. It simulates real-world network traffic and includes various attack types such as brute force, DoS/DDoS, web attacks, botnets, and infiltration. The dataset provides rich flow-level features extracted using CICFlowMeter, making it suitable for evaluating deep learning models.

3.1.3 Dataset Selection Justification

UNSW-NB15 and CICIDS2017 offer complementary perspectives on network traffic and attack behavior. Their combination enables a comprehensive and reliable evaluation of deep learning–based intrusion detection models. In order to provide a clear understanding of the experimental network environment, the network topology used for traffic generation and data collection is depicted in Figure 5.

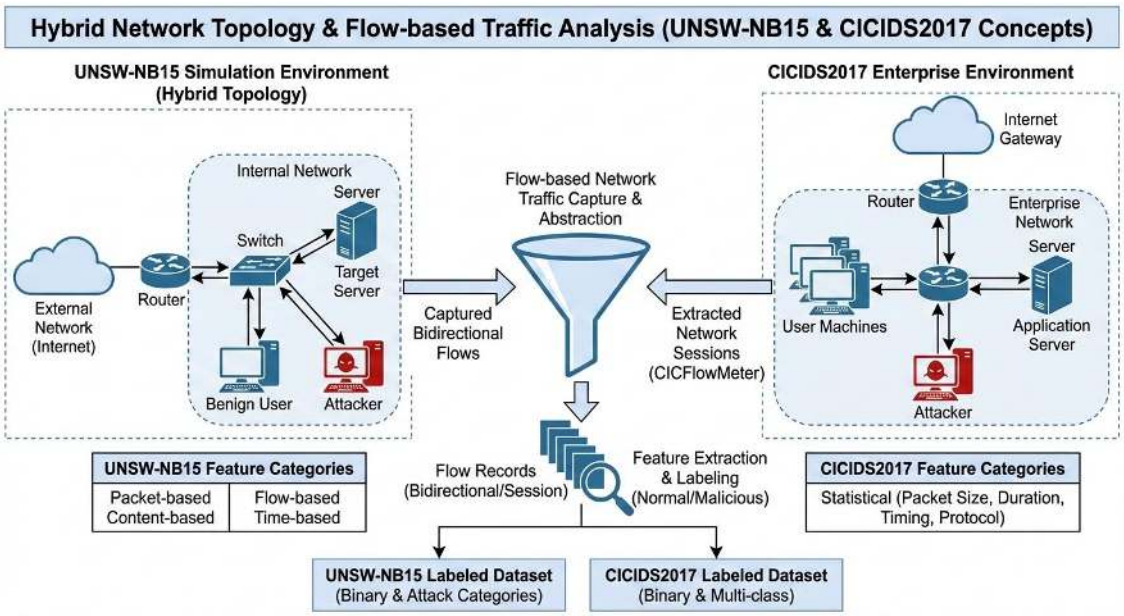


Figure 5: Hybrid Network topology & flow-based traffic Analysis (UNSW-NB15 & CIC-IDS2017 Concepts.)

3.2 Data Preprocessing

To ensure the reliability and effectiveness of the proposed intrusion detection system, a structured data preprocessing pipeline is applied to both UNSW-NB15 and CICIDS2017 datasets. This preprocessing process consists of five main stages: data cleaning, feature encoding, data normalization, class imbalance handling, and data splitting.

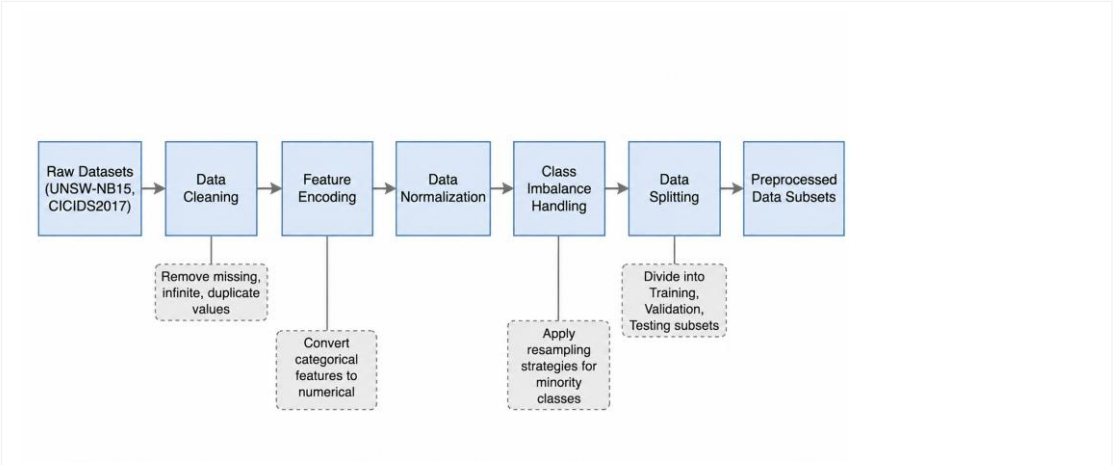


Figure 6: Data preprocessing pipeline for intrusion detection datasets, illustrating sequential steps from raw data through cleaning, feature encoding, normalization, class imbalance handling, and final data splitting into subsets for model development and evaluation.

This section presents in detail the preprocessing pipeline applied to the CIC-IDS2017 and UNSW-NB15 datasets for training our models in both binary and multiclass classification tasks. Data preprocessing is a crucial step that directly influences the quality and performance of intrusion detection models.

3.2.1 UNSW-NB15 Dataset Overview

The UNSW-NB15 dataset contains 40 numerical and three categorical features (proto, service, state) across 10 traffic categories. For binary classification, normal (65,100) and attack (115,271) samples yield a mild 1.77:1 imbalance. The multiclass setting is far more challenging, with the Normal class vastly exceeding Worms by 533:1. Additionally, byte-count features (sbytes, dbytes) reach values up to 10^9, making quantile-based normalization essential to ensure stable model training.

Table 3:UNSW-NB15 Preprocessing overview

Property	Value
Numerical features	40 (+ categorical: proto, service, state)
Binary target	'label' -> 0 = Normal, 1 = Attack
Multiclass target	'attack_cat' -> 10 categories
Attack types	Normal, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, Worms
Key challenge	Extreme scale disparity: sbytes / dbytes reach 10^9
Class imbalance (binary)	NORMAL 65,100 vs ATTACK 115,271 (ratio 1.77:1)
Class imbalance (multiclass)	Normal 65,100 vs Worms 122 (ratio 533:1)

3.2.2 Pipeline Order & Design Choices

Table 4: Representation of preprocessing steps.

Step	Operation	Choice	Justification
1	Missing values	drop / mean	Drop preserves statistical integrity; mean retains all rows
2	Categorical enc.	OneHot / Target	OneHot for low cardinality; Target encoding fit on train only.
3	Train/Val/Test split	Stratified 70/10/20	Preserves class proportions critical for 533:1 imbalance
4	Scaling	QuantileTransformer	Non-parametric; robust to log-normal distributions and 10^9 outliers

5	Oversampling	SMOTE	SMOTE for binary

3.2.3 Scaling Results on UNSW-NB15

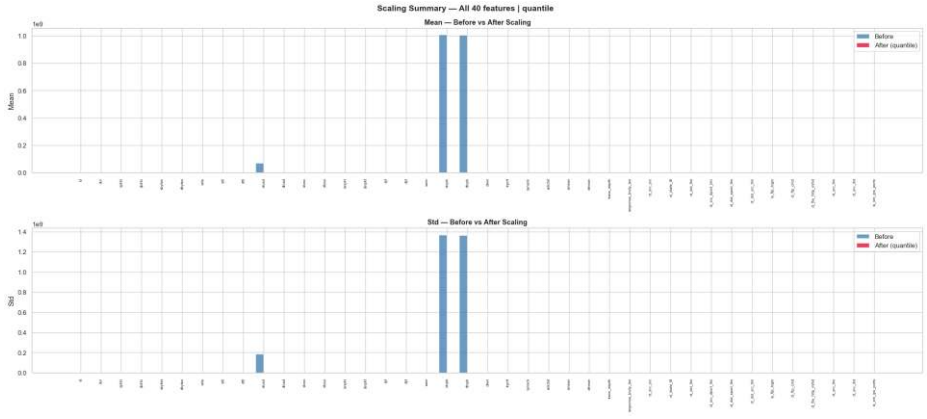


Figure 7: Scaling Summary for All 40 Features Using Quantile Normalization

This figure summarizes how all 40 features change after applying quantile normalization. The top panel shows the mean values, while the bottom panel shows the standard deviation, both before and after scaling. Before normalization, some features like *sbytes* and *dbytes* have extremely large values, reaching up to billions (10^9). After scaling (shown in red), all features are brought into a much smaller, consistent range centered around zero, making the data more stable and easier to work with.

In this work, for scaling method we have use quantiletransformer to handle outliers fixed skew because it is most recommended for IDS and optimal for network traffic features.

3.2.4 Feature Correlation Analysis

Correlation analysis serves two purposes in identifying informative features and detecting redundant ones.

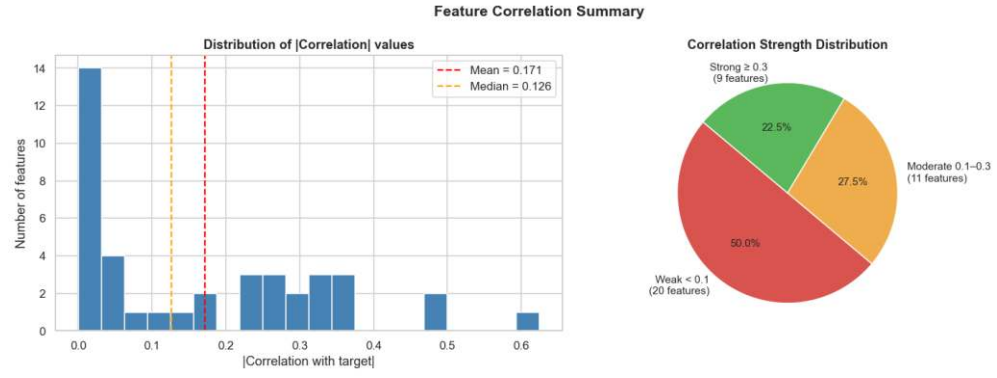


Figure 8: Feature correlation summary for binary classification.

Mutual Information-based Feature Selection for UNSW-NB15: This summary shows how features are selected based on how strongly they relate to the target variable. In total, 9 features stand out as highly informative (MI ≥ 0.3), while 11 others provide a moderate level of useful information and are also kept. On the other hand, 20 features with very low relevance (MI < 0.1) are considered for removal. In addition, 9 pairs of features are found to be highly similar to each other (correlation above 0.95), suggesting that one feature from each pair can be removed to reduce redundancy.

Redundant features: `sbytes, dbytes, sloss, dloss, dwin, ct_src_dport_ltm, ct_dst_src_ltm, ct_ftp_cmd, ct_srv_dst. Notably, sbytes` and `dbytes` are both dominant in magnitude and redundant with each other a double reason to handle them carefully.

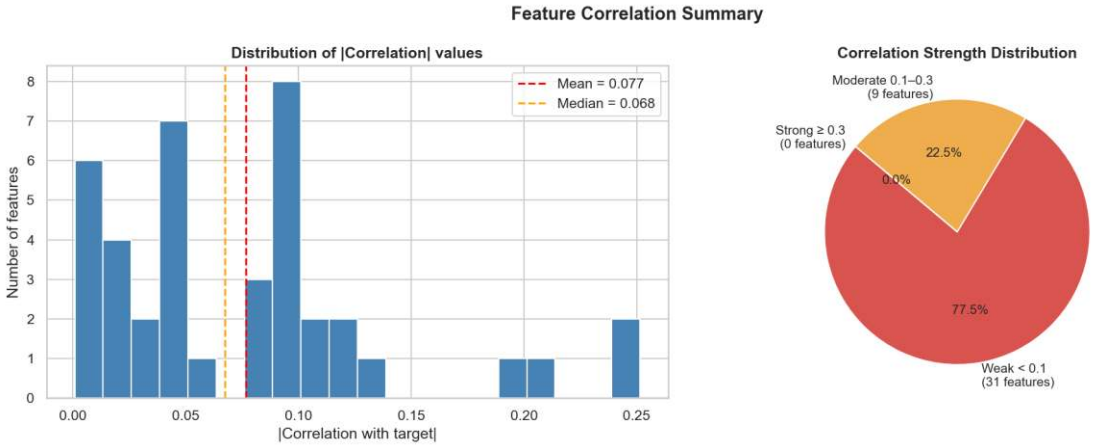


Figure 9: Feature correlation summary for multiclass classification.

In the multiclass setting, Pearson correlation with the encoded target yields a mean of 0.077 with no strongly correlated features (≥ 0.3), compared to 9 in the binary case. This degradation is expected: Pearson assumes a linear relationship with a continuous target, whereas multiclass labels are nominal integers with no ordinal meaning. Mutual Information is the appropriate metric for this configuration.

3.2.5 CIC-IDS2017 Dataset Overview

The CIC-IDS 2017 dataset consists of several features, totaling 78 numeric features plus the Label column. For binary classification, the Label target is used to differentiate between

benign and attack types. However, in a multiclass classification study, the Label is benign plus 14 subcategories with attack types such as: DoS, DDoS, PortScan, BruteForce, Bot, Infiltration, Web Attacks, Heartbleed, etc. The parameters between the benign and attack classes are so high that the ratio is approximately 4.1:1. In contrast, for multiclass classification, the ratio is approximately 1.6 million benign attacks versus a smallest attack class with fewer than 10 samples (ratio > 200,000:1). As a reminder, we do not need to encode these features because they are already numeric.

3.2.6 Pipeline Order & Design Choices

Our processing chain for CIC-IDS2017 dataset is built upon 5 different steps. First, we handle missing values if we're in training fold, we can perform mean imputation, but we need to calculate it on the training fold itself, in order not to reveal information to validation or testing fold. After handling missing values, we split data in a stratified 70% for training and validation, and 10/20% for validation. Scaling is performed by `QuantileTransformer`. This component helps to significantly reduce impact of large outliers as well as to mitigate impact of strong feature multi scale skewness typical for network traffic features. Next, we use SMOTE oversampling technique to balance binary class (×4.08 on attack class) but strictly on the training set. Finally, we encode categorical features we use OneHot encoding for low cardinality features, and Target Encoding for high cardinality ones. Crucial here is that Target Encoding is fitted only on training fold.

3.2.7 Feature Correlation Analysis

Correlation analysis was performed between each feature and the target variable using Pearson's absolute correlation coefficient. Two separate analyses were conducted for the multiclass and binary tasks, revealing markedly different distributions.

3.2.7.1 Multiclass Task Correlation

From the above analysis of Pearson correlation, it can be observed that for most of the 70 features of CIC-IDS2017 dataset, there is little correlation with multiclass label even when considered individually. The mean correlation for all 70 features is 0.076 and median correlation is 0.061. The table 13 gives an overview of the spread of correlation values obtained for all features. From table 13, it can be observed that only 1 feature (0.357) has correlation value higher than 0.3. 22 features (0.314) have moderate correlation value that is greater than 0.1 and less than 0.3. On the other hand, the value greater than 0.7 is obtained only for 5 features out of 70 which are 7% of total features. This is therefore considered as noise for our problem.

By the numbers, this feature looks pretty weak, with only a score of 0.04 and a negative Pearson feature importance. That said, Pearson only measures straight line relationships in data, and this feature may still have not trivial discriminative power even if it scores poorly, especially if that power comes from nonlinear relationships. As a result, this feature should not be eliminated solely based off the Pearson figure, and will likely have a different importance score when calculated by mutual information or by the feature importance from a tree.

3.2.7.2 Binary Task Correlation

When you look at the binary case it is a lot clearer and easier to understand than the setup. The average correlation is 0.151 with a median of 0.081. This means that the connection between the features and the NORMAL vs. ATTACK label is easier to see.

If you check Table 14 you will see a difference. This time 17 features are above the 0.3 threshold, which's a big change from the multiclass case where hardly any features stood out. There are also 14 features that're in a moderate range. So almost half of the features have some information.

Not everything gets better though. 55.7% Of the features still have very weak correlations, which are below 0.1.. Just because they are weak does not mean they are not useful. The binary task is just easier to do because when you put 14 attack types into one group the patterns become more obvious and the relationships are easier to find. The BENIGN vs. ATTACK label is simpler. That makes it easier to see the connections between the features and the BENIGN, vs. ATTACK label.

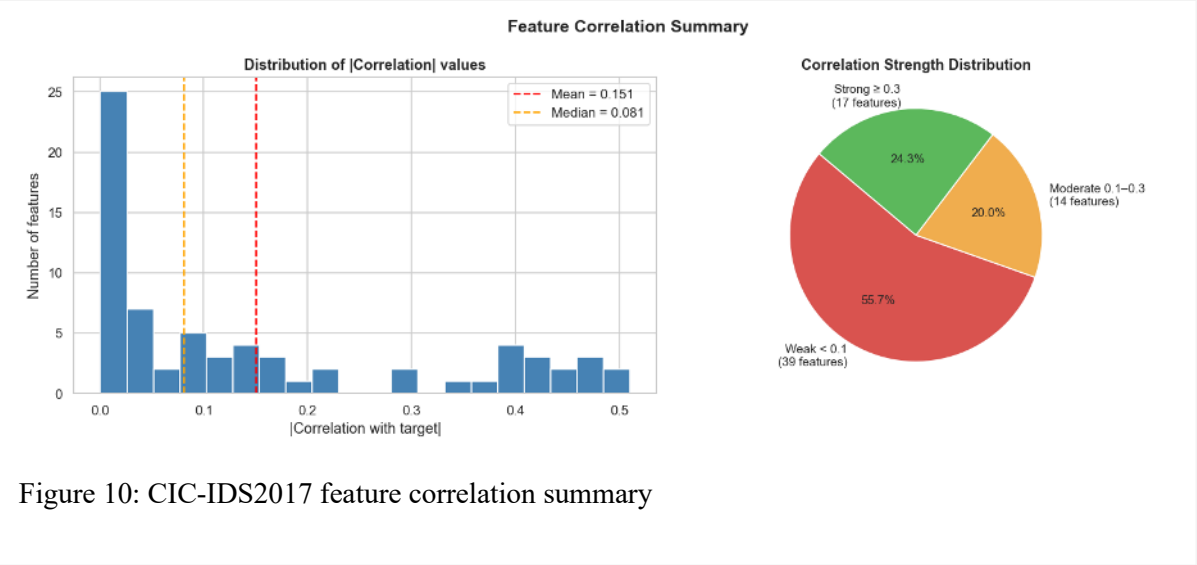


Figure 10: CIC-IDS2017 feature correlation summary

3.2.8 Data Scaling results on CIC-IDS2017

CIC-IDS2017 contains 70 features with highly heterogeneous scales. Several features exhibit means and standard deviations in the range of 10^7, while others remain below 1. Without scaling, the MLP would be dominated by high-magnitude features, causing gradient instability and poor convergence.

3.2.8.1 Scaler Comparison

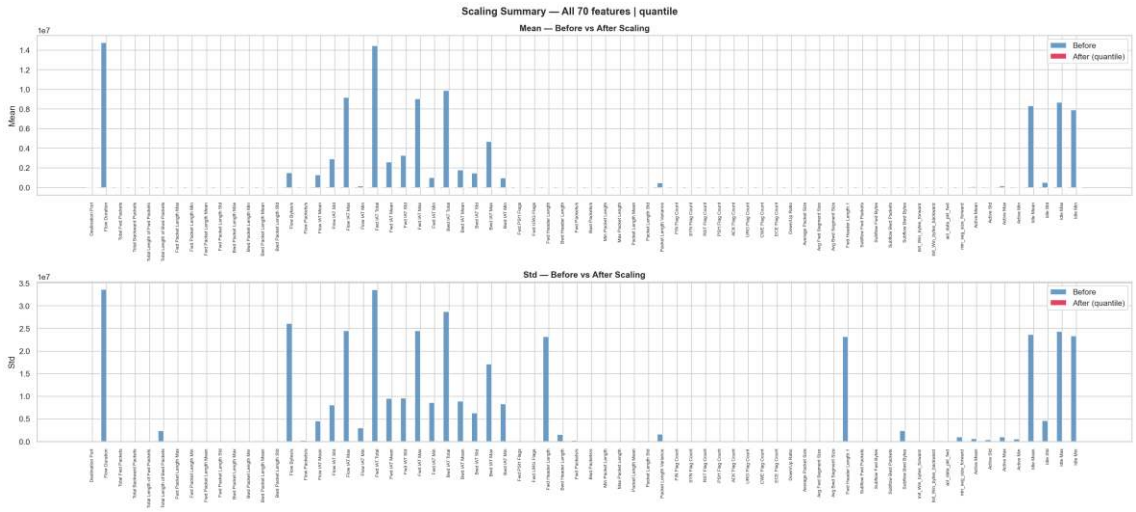


Figure 11: Multiclass scaling results

3.2.8.2 Observed Results

The Scaling Summary chart confirms that multiple features reach values around 10^7 before transformation (mean and std plots). After applying `QuantileTransformer`, all 70 features are compressed into a uniform [0, 1] distribution. The 'After (quantile)' bars are invisible on the 10^7 axis confirming the transformation is working correctly. Key high-magnitude features identified include packet length statistics, flow bytes per second, and duration-based features which commonly exhibit log-normal distributions in network traffic datasets.

3.2.9 Class Imbalance & Oversampling

3.2.9.1 Binary Task

The binary dataset presents a moderate imbalance with BENIGN dominating over ATTACK. SMOTE generates synthetic ATTACK samples by interpolating between k nearest neighbors in feature space until both classes are equalized.

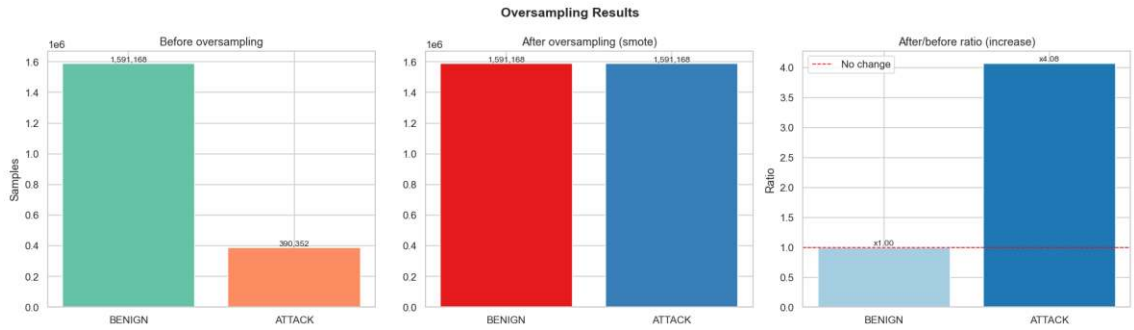


Figure 12: Oversampling after and before SMOTE

3.2.9.2 Multiclass Task Extreme Imbalance

The multiclass scenario reveals a severe imbalance characteristic of real-world network traffic. BENIGN represents the vast majority (~1.6M samples), while several attack categories contain only a handful of samples. This is one of the most challenging imbalance scenarios in the IDS literature.

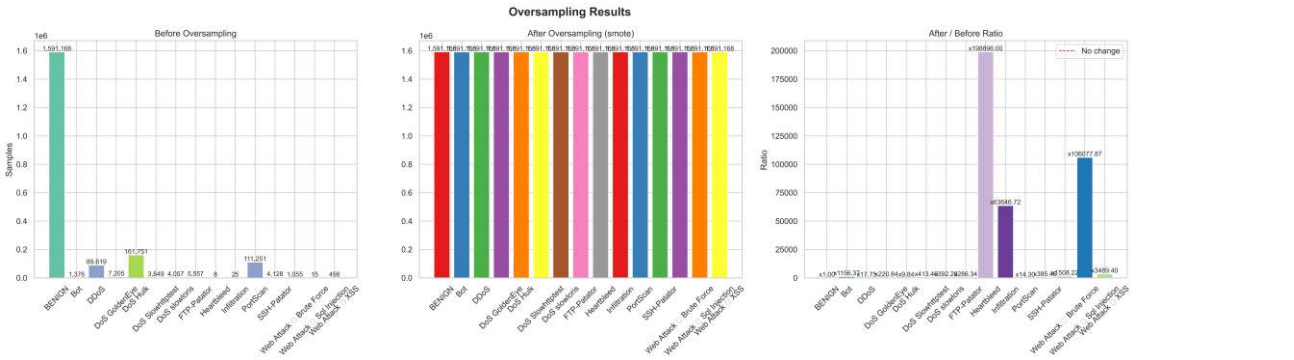


Figure 13: Oversampling after and before SMOTE.

3.3 Models

3.3.1 Overview and Motivation

In order to select architectures, three objectives were pursued simultaneously. The first one was to allow local spatial patterns on raw feature vectors to appear as useful information. The second one was to capture both sequential and bidirectional temporal relations in network flows. The third one was to find well known classical baselines for the chosen architectures in order to have a basis for comparison. A total of 10 different models were developed exploring 4 different architectural styles.

- One-Dimensional Convolutional Neural Networks (CNN) for spatial feature extraction.

Hybrid CNN-BiLSTM Networks: Hybrid models that combine hybrid strengths of local convolution modeling and bidirectional sequential modeling.

- Random Forest (RF) 'as an ensemble tree-based baseline';
- Various integer overheads.

In order to fairly evaluate each architecture, both the Normal vs. Attack (binary) and Multiple Classes (multiclass) settings are used as evaluation metrics, with the exception of classical ML models that are applied solely to the UNSW-NB15 dataset due to limitations in scale. All models are then evaluated under a unified and fair experimental protocol as described in Section 4.

3.3.2 CNN Architecture

3.3.2.1 Architecture Description

The CNN architecture employs 1D convolutions to process network traffic features as sequential data. The model consists of three convolutional blocks followed by fully connected layers for classification.

3.3.2.2 1D Convolution equation

For an input feature vector x and filter w, the 1D convolution is computed as:

$$y_i = \sum_{j=0}^{k-1} x_{i+j}.w_j + b \tag{16}$$

Here, x denotes the input signal (the network flow feature vector), w_j represents the filter weights (learned during training), j is the index over the kernel size, k is the kernel size (in our case, $k = 3$), i is the current position in the input sequence, y_i is the output value at position i, and b is the bias term. The base CNN for NIDS is constructed from stacked components, as shown in Figure 14.

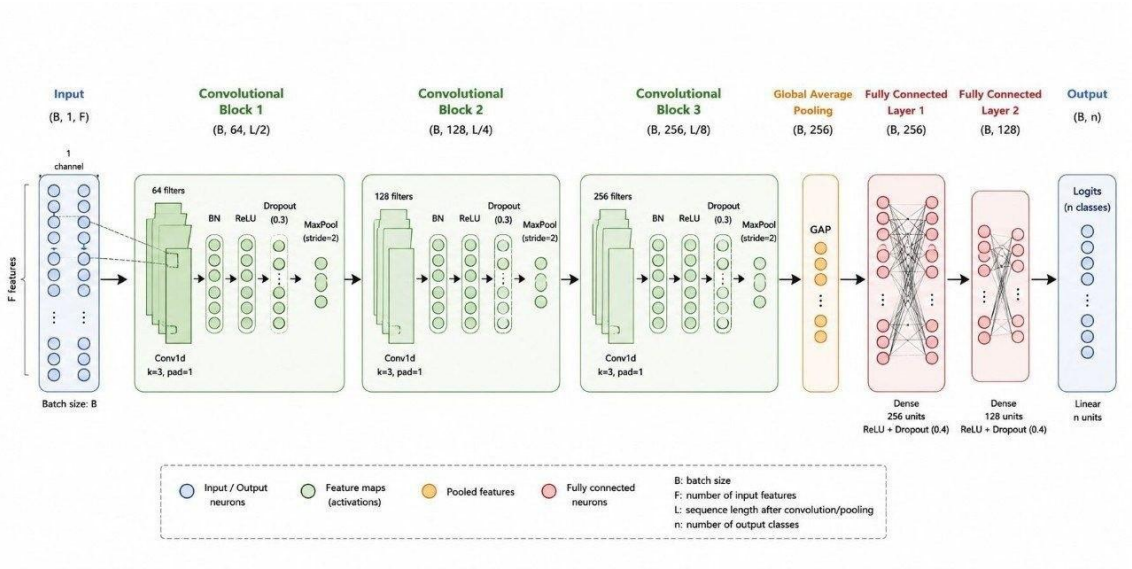


Figure 14: The proposed CNN architecture for this work.

3.3.3 CNN-BiLSTM Hybrid Architecture

In addition to the approach of using 1D CNNs for traffic classification, we also present a further deep learning method that specifically exploits features in network traffic: a hybrid CNN-BiLSTM architecture combining convolutional layers for feature extraction with Bidirectional Long Short-Term Memory (BiLSTM) networks that model sequence information of feature dimensions across multiple packets. While CNNs are capable of capturing local feature patterns in time for given features, they do not capture long-range dependencies. Recurrent units on the other hand are specifically designed to handle such long-range dependencies.

The proposed CNN-BiLSTM backbone network consists of two 1D convolutional blocks with residual structures, followed by a Bidirectional LSTM (BiLSTM) recurrent structure. Each convolutional block includes two 1D convolutional layers with increasing channel numbers (1–64–128) followed by a BatchNorm layer, ReLU activation, and MaxPooling layer, followed by a Dropout layer of 0.3. The features extracted from the feature maps of the last convolutional block are used as input for the BiLSTM recurrent structure with 128 hidden neurons in each direction, thus having 256 dimensions for the hidden states of each time step. In fact, traffic flow features have directional information that is asymmetric, where using past and future information together could capture relevant features.

The BiLSTM has tanh activation applied during cell state updates and sigmoid during the gating, following the standard architecture of an LSTM. A Dropout of 0.3 is also applied to the output of the LSTM. The last hidden state of the BiLSTM is then used as input into a

fully connected layer (256 units -> 128 units). This layer has ReLU activation and is regularized with a Dropout of 0.3. The output of this is then passed to the final output layer where Softmax activation is applied to output a probability distribution over the required number of classes.

Compared with single CNN1D model, the CNN-BiLSTM model exploits relational modeling to take advantage of inter-feature sequential structures where feature ordering conveys semantic meaning, especially for engineered traffic features that are aggregated from packet-level statistics. This is a state-of-the-art practice in recent NIDS literature.

3.3.4 Random Forest

Random Forest (RF) is an ensemble learning method that consists of a collection of decision trees, where the classification of an input, if performed by the individual trees, results in a distribution of outputs with the final class being the most common outcome. The variance reduction mechanism of RF relies on bootstrap sampling and feature randomness, and is particularly effective in handling high dimensionality.

For this test RF was applied to the binary classification task on UNSW-NB15. This used the Mutual Information (MI) feature selection method top-30 to try to avoid the noise from irrelevant features. Additionally, prior to training SMOTE was used to try to mitigate class imbalance. While training time was somewhat high at 5,069 seconds, accuracy and F1-Score were high at 0.9854 and 0.9885 respectively, showing RF to be a strong baseline.

3.3.5 XGBoost

Extreme Gradient Boosting, short XGBoost, is a scalable and distributed Gradient Boosting framework designed to be extremely efficient both in terms of computational resource usage as well as to resist overfitting. The algorithm constructs an ensemble of numerous decision trees in a stage-wise fashion and in each node, splits the data based on feature values. The regularized objective function used in XGBoost uses both L1 and L2 regularization as well as takes advantage of second-order gradient statistics.

The performance of XGBoost is evaluated on both the binary and multiclass classification tasks on UNSW-NB15. For the multiclass task (10 classes), a custom version of Focal Loss is implemented as a custom objective function to further address the class imbalance issue, along with SMOTE. Feature encoding is performed using `TargetEncoder` fitted only on the training data to prevent any potential target leakage. The most informative features are selected using Mutual Information (MI) top-30 feature selection, but this time on the entire training fold. The models were trained within a matter of seconds on a GPU, specifically in 203 seconds for the binary task and 30.9 seconds for the best multiclass model.

3.3.6 Training Configuration

A unified and reproducible training protocol is adopted across all deep learning experiments to ensure fair comparisons. The key hyperparameters, regularization strategies, and optimization settings are summarized below.

3.3.7 Hyperparameters

Table 5 presents the complete hyperparameter configuration for all eight deep learning model configurations (2 architectures × 2 tasks × 2 datasets). All models are trained using the Adam optimizer with a learning rate of 1e-3. Batch sizes are set to 512 for CIC-IDS2017 (larger dataset) and 256 for UNSW-NB15, with a maximum of 100 epochs per fold. Early stopping with a patience of 5 epochs on validation loss is applied universally to prevent overfitting and reduce unnecessary computation.

Table 5: Hyperparameter configuration for all deep learning models

Dataset	**Architecture**	**Task**	**Activations**	**Dropout**	**Optimizer & LR**	**Loss**	**Epochs / Batch**	**Early Stop**
CIC-IDS2017	CNN1D (3 Conv1d blocks 1→64→128→256 + GAP + 2 FC 256→128→2)	Binary	ReLU (conv+FC) Softmax (output via CE)	0.3 conv 0.4 FC	Adam lr=1e-3	Focal Loss (γ=2, α=0.25)	100 / 512	patience=5
CIC-IDS2017	CNN-BiLSTM (2 Conv1d 1→64→128 + BiLSTM 128→256 + FC 256→128→2)	Binary	ReLU (conv+FC) tanh/sigmoid (LSTM) Softmax (output)	0.3 conv 0.3 LSTM 0.3 FC	Adam lr=1e-3	Focal Loss (γ=2, α=0.25)	100 / 512	patience=5
CIC-IDS2017	CNN (3 Conv1d blocks 1→64→128→256 + GAP + 2 FC 256→128→15)	Multiclass (15 classes)	ReLU (conv+FC) Softmax (output via CE)	0.3 conv 0.4 FC	Adam lr=1e-3	Focal Loss (γ=2.0)	100 / 256	patience=5

CIC-IDS2017	CNN-BiLSTM (2 Conv1d 1→64→128 + BiLSTM 128→256 + FC 256→128→15)	Multiclass (15 classes)	ReLU (conv+FC) tanh/sigmoid (LSTM) Softmax (output)	0.3 conv 0.3 LSTM 0.3 FC	Adam lr=1e-3	Focal Loss (γ=2.0)	100 / 128	patience=5
UNSW-NB15	CNN (3 Conv1d blocks 1→64→128→256 + GAP + 2 FC 256→128→2)	Binary	ReLU (conv+FC) Softmax (output via CE)	0.3 conv 0.4 FC	Adam lr=1e-3	Focal Loss (γ=2, α=0.25)	100 / 256	patience=5
UNSW-NB15	CNN-BiLSTM (2 Conv1d 1→64→128 + BiLSTM 128→256 + FC 256→128→2)	Binary	ReLU (conv+FC) tanh/sigmoid (LSTM) Softmax (output)	0.3 conv 0.3 LSTM 0.3 FC	Adam lr=1e-3	Focal Loss (γ=2, α=0.25)	100 / 256	patience=5
UNSW-NB15	CNN (3 Conv1d blocks 1→64→128→256 + GAP + 2 FC 256→128→10)	Multiclass (10 classes)	ReLU (conv+FC) Softmax (output via CE)	0.3 conv 0.4 FC	Adam lr=1e-3	Focal Loss (γ=2.0)	100 / 256	patience=5
UNSW-NB15	CNN-BiLSTM (2 Conv1d 1→64→128 + BiLSTM 128→256 + FC 256→128→10)	Multiclass (10 classes)	ReLU (conv+FC) tanh/sigmoid (LSTM) Softmax (output)	0.3 conv 0.3 LSTM 0.3 FC	Adam lr=1e-3	Focal Loss (γ=2.0)	100 / 256	patience=5

3.3.8 Loss function

Standard cross-entropy loss is insufficient for heavily imbalanced NIDS datasets, as it assigns equal importance to all samples regardless of class frequency. We therefore adopt **Focal Loss**, originally proposed by Lin in (Lin et al., 2017) for dense object detection, across all deep learning models and the XGBoost multiclass experiment.

Focal Loss modifies the standard binary/multiclass cross-entropy by introducing a modulating factor $(1 - p_t)^{\gamma}$ that down-weights the contribution of easy, well-classified examples and focuses learning on hard, misclassified samples. The formulation is:

$$FL(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log (p_t) \tag{17}$$

Where:

$$p_t = \begin{cases} p & if\ y = 1 \\ 1 - p & if\ y = 0 \end{cases} \tag{18}$$

$$\alpha_t = \begin{cases} \alpha & if\ y = 1 \\ 1 - \alpha & if\ y = 0 \end{cases} \tag{19}$$

3.3.9 Fold Stratified Cross-Validation

All deep learning experiments are conducted using 5-fold stratified cross-validation to ensure robust and unbiased performance estimates. Stratification preserves the class distribution across all folds, preventing any fold from containing a disproportionate representation of minority classes.

For each fold, the data is split into training (70%) and validation (10%) subsets. The preprocessing pipeline including feature scaling (Quantile Transformer) and SMOTE oversampling is fit exclusively on the training portion of each fold and then applied to the validation subset using the fitted parameters. This ensures a strictly leak-free evaluation. Final performance metrics are reported as the mean and standard deviation across the five folds.

For classical ML models (RF and XGBoost on UNSW-NB15), a fixed stratified 70/10/20 split is used (train/validation/test) rather than cross-validation, due to the higher computational cost of tree ensemble methods on large datasets.

3.3.10 Summary: Classical ML Configuration on UNSW-NB15

For reference, Table 7 summarizes the configuration and best results obtained for the classical ML baselines (Random Forest and XGBoost) on UNSW-NB15, providing context for the comparison presented in Chapter 4.

Table 6: Summary of classical ML configuration and results on UNSW-NB15

Aspect	**Binary (RF vs XGB)**	**Multiclass (XGB)**
Task	Normal vs. Attack (Binary)	10 attack categories (Multiclass)
Best Accuracy	0.9854	0.8416
Best F1-Score	0.9885 (weighted / binary)	0.8496 (weighted)
F1-Macro	N/A	0.6140
Training Time	RF: 5,069 s | XGB: 203 s	XGB: 30.9 s (best model)
Class Balancing	SMOTE (2 classes)	SMOTE k=3 (10 classes)
Loss Function	Binary logistic	Focal Loss (γ=2.0, α=0.25)
Encoding	LabelEncoder	TargetEncoder (leak-free)
Feature Selection	MI top-30 (full train set)	MI top-30 (train only)

Chapter 4

4 Experimental results and Analysis

4.1 Experimental setup

Before any model is trained, the raw data goes through a consistent preprocessing pipeline. Missing values are handled through mean imputation, and infinite values which appear in the CIC-IDS2017 dataset due to division-by-zero operations in flow statistics are replaced with NaN prior to imputation. Feature scaling is performed using a QuantileTransformer, which maps each feature to a uniform distribution and is particularly well-suited for the heavy-tailed, skewed distributions typical of network traffic data.

The data is split in a stratified manner to preserve class proportions: 70% for training, 10% for validation, and 20% for testing. To address the class imbalance, present in both datasets, SMOTE (Synthetic Minority Over-sampling Technique) is applied within each fold during cross-validation, ensuring that no information from the test fold leaks into the oversampling process. All models are trained with Focal Loss (γ=2.0, α=0.25), which down-weights well-classified examples and focuses learning on the harder, minority-class samples.

4.2 Model Training and Evaluation

Each model is evaluated using 5-fold stratified cross-validation. Early stopping is applied based on the validation loss with a patience value that varies slightly by configuration, preventing overfitting while allowing sufficient convergence. The best-performing fold selected by F1-score on the validation set is then used to evaluate the model on the held-out test set.

For binary classification, the default decision threshold of 0.5 is replaced by an optimal threshold derived from Youden's J index ($J = TPR - FPR$), computed on the validation set. This threshold maximizes the balance between sensitivity and specificity, which is especially valuable in security contexts where both missed attacks and false alarms carry operational costs.

The reported metrics include accuracy, precision, recall, F1-score, false positive rate (FPR), ROC-AUC, average precision (PR-AUC), detection time per sample, and throughput.

5-fold cross-validation is applied only to the 80% training + validation portion of the data. The 20% test set is kept completely separate and is not used at any stage during training or model selection. Within this 80% portion, each fold constitutes approximately 10% of the total data as the validation set and approximately 70% as the training set. This structure is fully consistent with the 70% / 10% / 20% data split ratio specified in the thesis.

The cross-validation method applied in this study can be described in detail as follows and the dataset is first divided into 5 subsets (folds), and in each iteration:

- One fold is used as the validation set
- The remaining 4 folds are used for training the model
- The model's performance is evaluated on the validation set

Table 7: 5-Fold Cross-Validation Split Scheme.

Iteration	Training Set	Validation Set
Fold 1	Folds 2, 3, 4, 5	Fold 1
Fold 2	Folds 1, 3, 4, 5	Fold 2
Fold 3	Folds 1, 2, 4, 5	Fold 3
Fold 4	Folds 1, 2, 3, 5	Fold 4
Fold 5	Folds 1, 2, 3, 4	Fold 5

Here is the faithful academic translation:

- **Fold 1:** The model was trained on 80% of the data and tested on the first fold (20%). The obtained evaluation metrics correspond to this test.
- **Fold 2:** The same procedure was applied; however, this time the second fold was used as the test set.
- **...**
- **Fold 5:** The fifth fold was used as the test set.

4.3 Binary Classification Results

4.3.1 CIC-IDS2017 Binary Classification (NORMAL vs ATTACK)

CNN: CIC-IDS2017 Binary

The CNN model achieves strong performance on the CIC-IDS2017 binary task, demonstrating both high discriminative capacity and computational efficiency. Across the five folds, the model maintains a mean accuracy of **99.75%,** reflecting near-perfect separation between benign and attack traffic. Performance Summary of CNN1D Binary (CIC-IDS2017) is listed on the table below:

Table 8: Five-Fold Cross-Validation Results of the Proposed CNN: CIC-IDS2017 Binary Model.

Fold	Accuracy	Precision	Recall	F1-Score	FPR	ROC-AUC	PR-AUC	Detection Time (ms/sample)	Throughput (samples/s)
1	0.9972	0.9878	0.9983	0.9930	0.0030	0.9999	0.9983	0.0097	103237.7160
2	0.9974	0.9885	0.9984	0.9934	0.0029	0.9999	0.9984	0.0102	97958.1255
3	0.9975	0.9876	0.9997	0.9936	0.0031	0.9999	0.9997	0.0097	103545.1541
4	0.9971	0.9870	0.9985	0.9927	0.0032	0.9999	0.9985	0.0099	101174.0899
5	0.9983	0.9930	0.9986	0.9958	0.0017	1.0000	0.9986	0.0094	106071.0436

Table 9: Summary of CNN performance on CIC-IDS2017 (binary, mean over 5 folds)

Metric	Value
Accuracy (mean)	0.9975
Precision: ATTACK	0.9888
Recall / Detection Rate (TPR)	0.9987
F1-Score: ATTACK	0.9937
FPR	0.0019 (mean)
ROC-AUC	0.9999
PR-AUC	0.9998
Detection Time	0.0098 ms/sample
Throughput	≈ 102,397 samples/sec

Selected best model: Fold 5 (highest F1-Score) decision threshold: 0.4679

Table 10: Test Set Results (Binary CNN – CIC-IDS2017).

Metric	**Value**
Accuracy	0.9982
Precision (Attack)	0.9919
Recall (Attack)	0.9992
F1-Score (Attack)	0.9956
False Positive Rate (FPR)	0.0020
Detection Rate	0.9992
ROC-AUC	0.9999
Average Precision	0.9998
Average Detection Time	0.0096 ms/sample
Throughput	104,045 samples/s

The confusion matrix reveals that among the 111,551 attack samples in the test set, the model correctly identifies 111,465 of them (99.9%), missing only 86 a remarkable detection rate that confirms the model's capacity to generalize over highly imbalanced traffic distributions. Similarly, among benign samples, only 909 out of 454,721 are incorrectly flagged as attacks, yielding a false positive rate of 0.002.

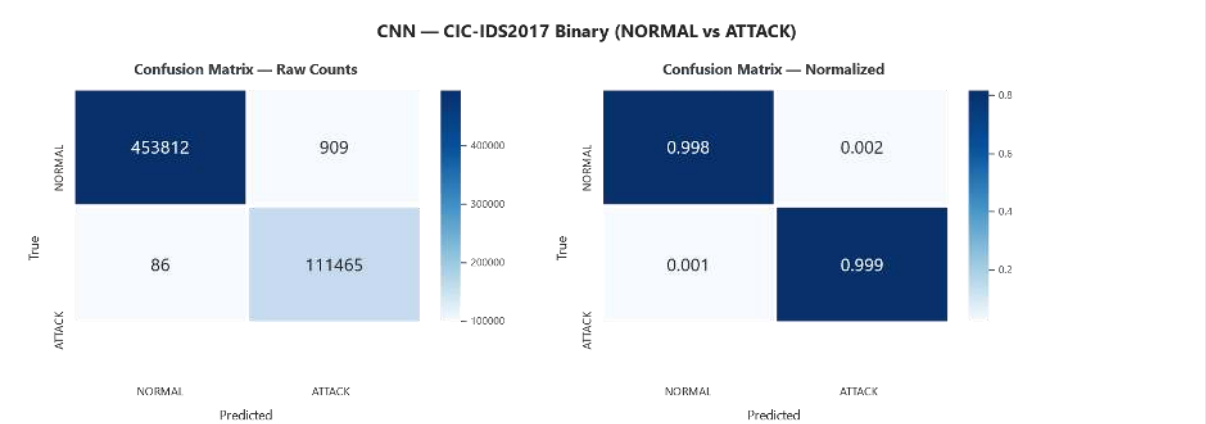


Figure 15: Confusion matrices (raw counts and normalized) for CNN on CIC-IDS2017 binary

The training curves confirm rapid and stable convergence. The model reaches an accuracy above 0.995 within the first 5 epochs and plateaus smoothly around 0.9975 by epoch 20, with training and validation curves remaining tightly aligned throughout, a clear indicator of the absence of overfitting. The loss curve mirrors this behavior, decreasing sharply in the early epochs before stabilizing at a very low value (≈ 0.0005), with negligible divergence between train and validation.

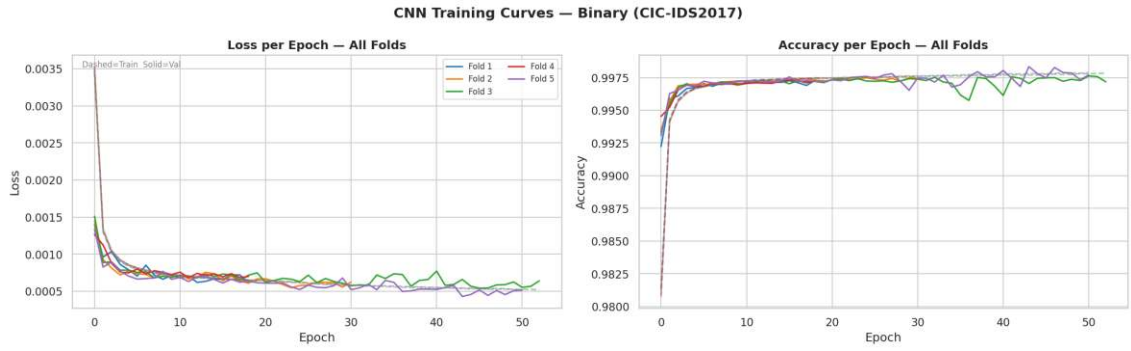


Figure 16: Training and validation learning curves (accuracy and loss) for CNN1D on CIC-IDS2017 binary classification.

The ROC curve further confirms the model's discriminative power: the AUC reaches 1.0000 on the validation set, with the optimal operating threshold identified at 0.468 via the Youden's J statistic. This indicates that the model separates the two classes almost perfectly in the probability space. The per-fold metrics bar charts additionally confirm the stability of the results across folds, with negligible variance in all metrics.

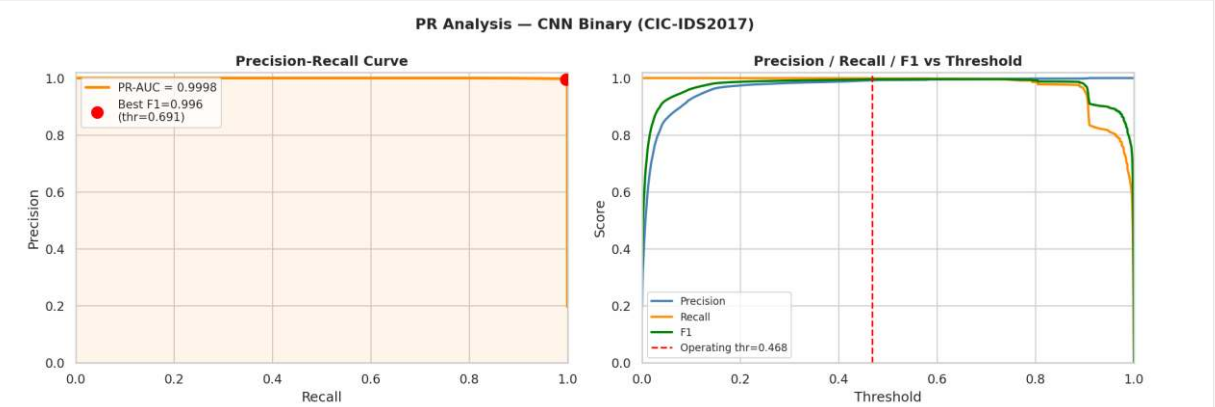


Figure 17: ROC curve, Youden's J threshold analysis, and per-fold metric distributions for CNN on CIC-IDS2017 binary classification

In terms of operational efficiency, the CNN processes each sample in approximately 0.0098 ms, with a throughput exceeding 102,000 samples per second making it well suited for real-time deployment scenarios where both accuracy and speed are critical requirements.

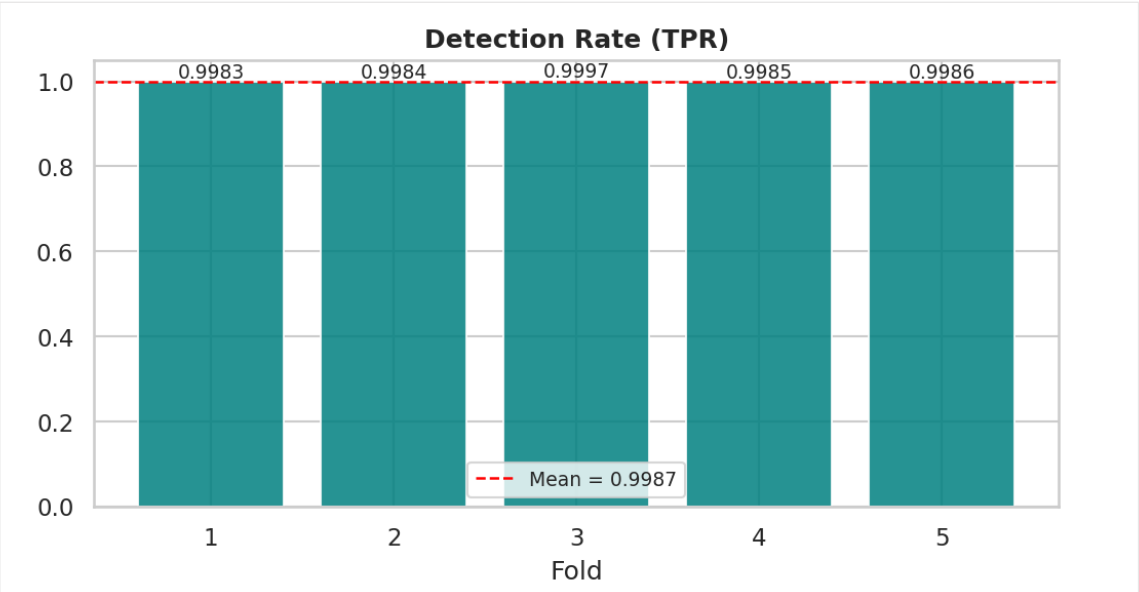


Figure 18: Per-fold Detection Rate (TPR) and Detection Time for CNN1D on CIC-IDS2017 binary classification.

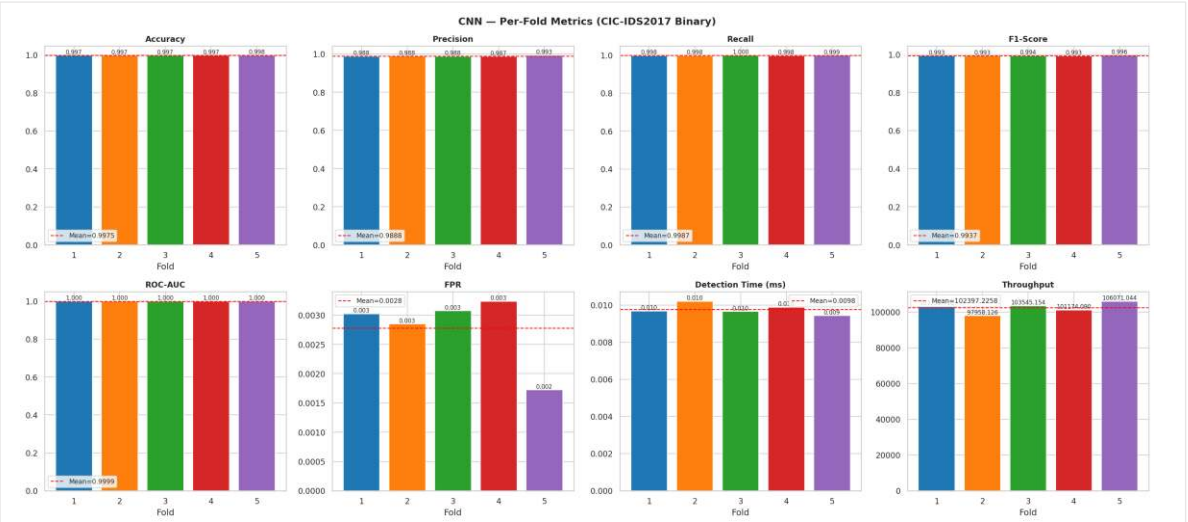


Figure 19: Per-Fold Metrics (Accuracy, Precision, Recall, F1, FPR, ROC-AUC, Detection Time, Throughput) CNN-Binary CIC-IDS2017

4.3.2 CNN-BiLSTM: Binary Classification (NORMAL vs ATTACK)

On the CIC-IDS2017 binary task, this hybrid model delivers performance that is highly competitive with the CNN1D baseline, and in some metrics slightly superior.

Table 11: Five-Fold Cross-Validation Results of the Proposed CNN-BiLSTM: CIC-IDS2017 Binary Model.

Fold	Accuracy	Precision	Recall	F1-Score	FPR	ROC-AUC	Detection Rate	Detection Time (ms)	Throughput (samples/s)
1	0.9969	0.9866	0.9978	0.9922	0.0033	0.9999	0.9978	0.0458	21,845.2172
2	0.9968	0.9860	0.9978	0.9918	0.0035	0.9999	0.9978	0.0459	21,771.7182
3	0.9968	0.9856	0.9981	0.9918	0.0036	0.9999	0.9981	0.0458	21,852.2420
4	0.9964	0.9838	0.9984	0.9910	0.0040	0.9999	0.9984	0.0458	21,853.1849
5	0.9969	0.9863	0.9982	0.9922	0.0034	0.9999	0.9982	0.0458	21,818.0437

Table 12: Summary of CNN-BiLSTM performance on CIC-IDS2017 (binary, mean over 5 folds)

Metric	Value
Accuracy (mean)	0.9959
Precision - ATTACK	0.9857
Recall / Detection Rate (TPR)	0.9981
F1-Score - ATTACK	0.9918
FPR	0.0036 (mean)
ROC-AUC	0.9999
PR-AUC	0.9996

Detection Time	0.0458 ms/sample
Throughput	≈ 21,828 samples/sec

Selected best model: Fold 5 (highest F1-Score) d**ecision threshold:** 0.5234

Table 13:Test Set Results (CNN-BiLSTM Binary Classification – CIC-IDS2017).

Metric	**Value**
Accuracy	0.9968
Precision (Attack)	0.9865
Recall (Attack)	0.9976
F1-Score (Attack)	0.9920
False Positive Rate (FPR)	0.0034
Detection Rate	0.9976
ROC-AUC	0.9999
PR-AUC	0.9996
Average Detection Time	0.0464 ms/sample
Throughput	21,548 samples/s

The confusion matrix confirms that the model correctly classifies 453,296 benign samples out of 454,822 (99.7%), and detects 111,314 attack samples out of 111,578 (99.8%). The 264 missed attacks and 1,526 false alarms represent a minor degradation compared to the CNN, partially attributable to the added complexity of the recurrent component and the resulting sensitivity to class boundaries.

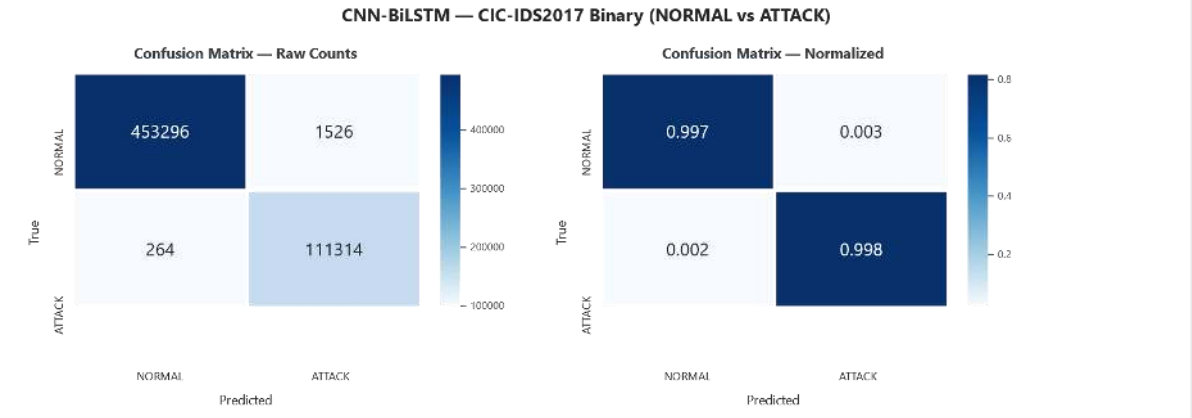


Figure 20: Confusion matrices (raw counts and normalized) for CNN-BiLSTM on CIC-IDS2017 binary classification.

The training dynamics reveal a different convergence profile compared to the CNN. With early stopping activated, the model converges in approximately 10 epochs rather than 50.

Accuracy rises sharply to approximately 0.996 within the first two epochs, and the loss drops from 0.0025 to below 0.001 very rapidly. However, a slight divergence between train and validation loss is observed from epoch 7 onwards, suggesting the onset of mild overfitting a known limitation of architectures with recurrent components trained on datasets with limited temporal structure at the sample level.

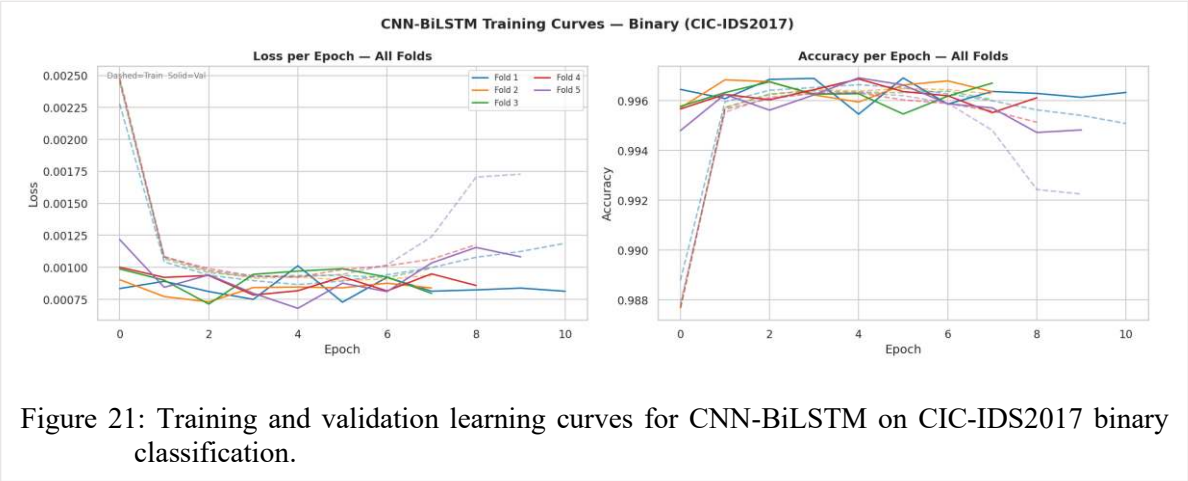


Figure 21: Training and validation learning curves for CNN-BiLSTM on CIC-IDS2017 binary classification.

Despite this, the ROC-AUC remains at 0.9999, and the Precision-Recall AUC reaches 0.9996. The optimal threshold is identified at 0.523 via the Youden's J criterion, close to the standard 0.5, indicating well-calibrated probability outputs. Per-fold results are highly consistent, with standard deviations below 0.002 across all key metrics.

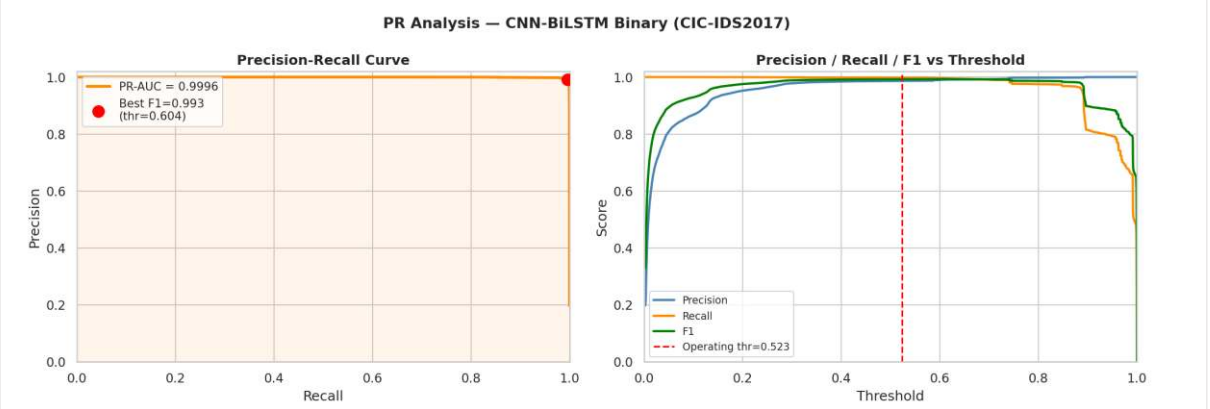


Figure 22: ROC curve, Precision-Recall curve, Youden's J, and per-fold metric distributions for CNN-BiLSTM on CIC-IDS2017 binary classification

The primary trade-off of the CNN-BiLSTM lies in its computational cost. The model processes each sample in 0.0458 ms approximately 4.7 times slower than the CNN and achieves a throughput of roughly 21,800 samples per second. While this remains sufficient for many monitoring scenarios, it represents a meaningful disadvantage in high-throughput network environments.

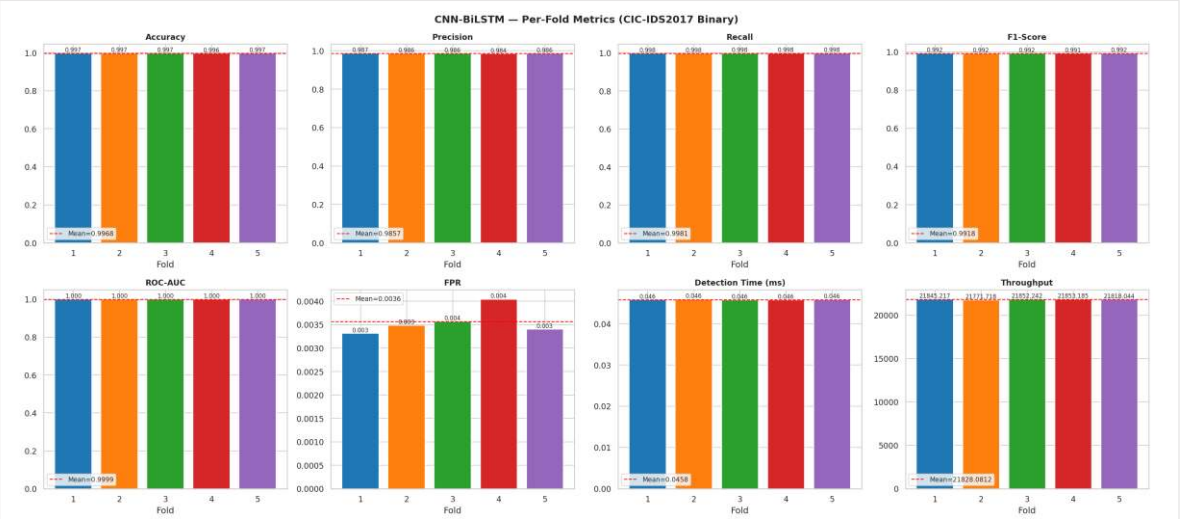


Figure 23: Per-Fold Metrics (Accuracy, Precision, Recall, F1, FPR, ROC-AUC, Detection Time, Throughput) CNN-BiLSTM Binary CIC-IDS2017

4.3.3 UNSW-NB15 Binary Classification (NORMAL vs ATTACK)

CNN: UNSW-NB15 Binary Class

On the UNSW-NB15 binary task, the CNN model delivers solid but slightly lower performance compared to CIC-IDS2017, reflecting the greater inherent difficulty of this dataset. The mean accuracy across the five folds stands at **95.23%,** with performance remaining stable and consistent throughout all folds.

Table 14: Five-Fold Cross-Validation Results of the Proposed CNN: UNSW-NB15 Binary Model.

Fold	Accuracy	Precision	Recall	F1-Score	FPR	ROC-AUC	Detection Rate	Detection Time (ms)	Throughput (samples/s)
1	0.9509	0.9837	0.9388	0.9607	0.0276	0.9925	0.9388	0.0119	84,280.5130
2	0.9540	0.9855	0.9420	0.9632	0.0246	0.9935	0.9420	0.0163	61,515.0534
3	0.9515	0.9795	0.9439	0.9614	0.0350	0.9922	0.9439	0.0117	85,501.8999
4	0.9520	0.9859	0.9383	0.9615	0.0238	0.9935	0.9383	0.0120	83,462.6943
5	0.9520	0.9837	0.9405	0.9616	0.0276	0.9927	0.9405	0.0117	85,514.1755

Table 15: Summary of CNN1D performance on UNSW-NB15 (binary, mean over 5 folds)

Metric	Value
Accuracy (mean)	0.9523
Precision - ATTACK	0.9638
Recall / Detection Rate (TPR)	0.9407
F1-Score - ATTACK	0.9617
FPR	0.0317 (mean)
ROC-AUC	0.9942
PR-AUC	0.9964
Detection Time	0.0127 ms/sample
Throughput	≈ 80,055 samples/sec

Selected best model: Fold 2 (highest F1-Score) d**ecision threshold:** 0.4708.

Table 16: Test Set Results (Binary CNN – UNSW-NB15).

Metric	Value
Accuracy	0.9562
Precision (Attack)	0.9815
Recall (Attack)	0.9493
F1-Score (Attack)	0.9652
False Positive Rate (FPR)	0.0316
Detection Rate	0.9493
ROC-AUC	0.9935
Average Precision	0.9964
Average Detection Time	0.0122 ms/sample
Throughput	82,168 samples/s

The confusion matrix shows that from a total of 32,882 attack samples, the CNN correctly detects 31,216 (94.9%), while misclassifying 1,666 as benign. Conversely, 587 benign samples are incorrectly labelled as attacks, out of 18,574 total corresponding to a false positive rate of approximately 3.2%. This higher FPR compared to CIC-IDS2017 is consistent with the more challenging nature of UNSW-NB15, where benign and attack traffic distributions overlap more significantly.

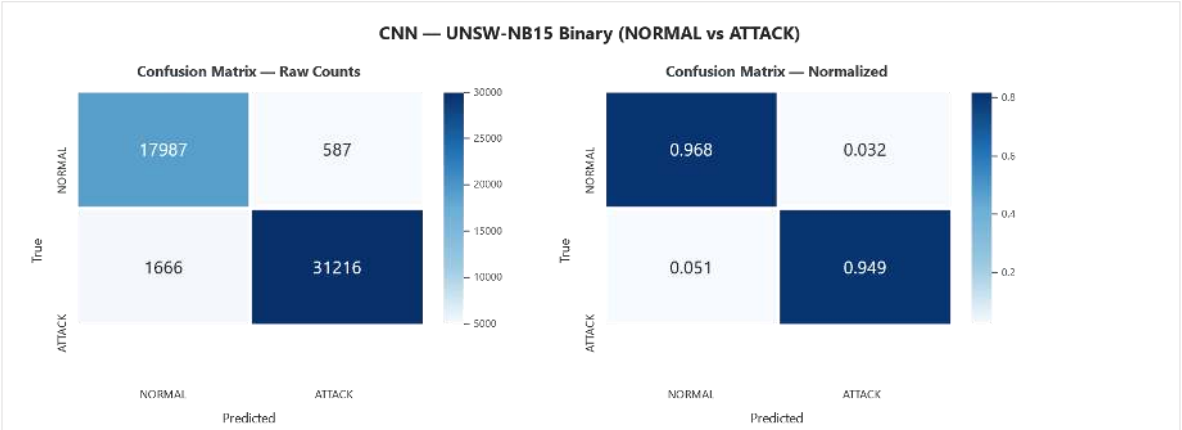


Figure 24: Confusion matrices (raw counts and normalized) for CNN on UNSW-NB15 binary classification.

The training curve is smooth and progressive. Accuracy rises steadily from 0.89 to approximately 0.955 over 50 epochs, with the validation curve closely tracking the training curve throughout, suggesting good generalization. The loss decreases from around 0.016 to a stable plateau near 0.007, with no notable divergence between training and validation loss an encouraging sign given the dataset's complexity.

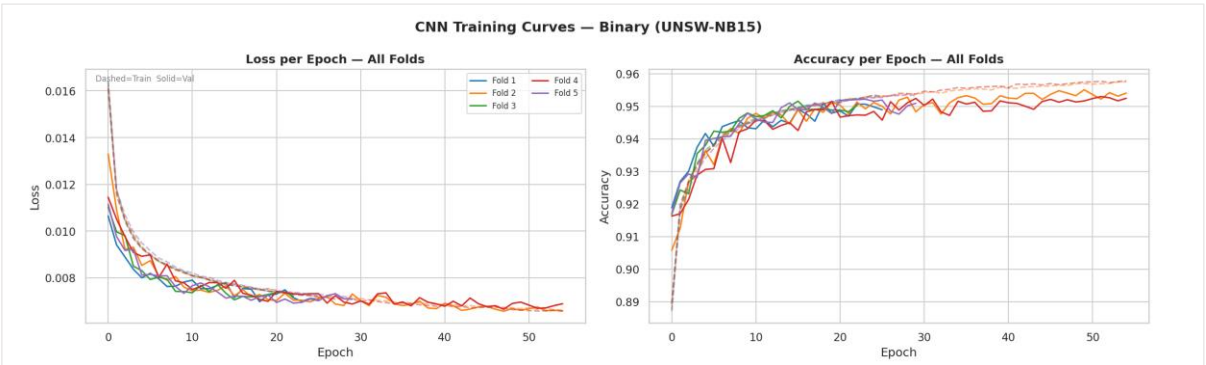


Figure 25: Training and validation learning curves for CNN1D on UNSW-NB15 binary classification.

The ROC-AUC reaches 0.9942 on the validation set, and the Precision-Recall AUC stands at 0.9964 both high values that confirm the model's strong discriminative ability despite the more balanced and complex class structure. The optimal threshold is set at 0.471, identified via Youden's J. Per-fold metrics exhibit very low variance, confirming the robustness and reproducibility of the results.

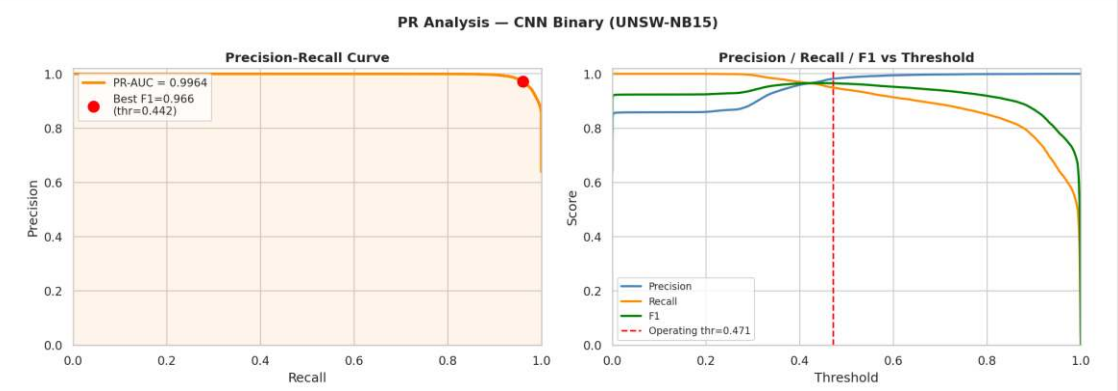


Figure 26:ROC curve, Precision-Recall curve, Youden's J, and per-fold metric distributions for CNN on UNSW-NB15 binary classification

From an operational standpoint, the CNN processes each sample in **0.0127 ms** on average, with a throughput of approximately 80,000 samples per second a strong result that positions the model as a viable candidate for real-time deployment even on this more demanding dataset.

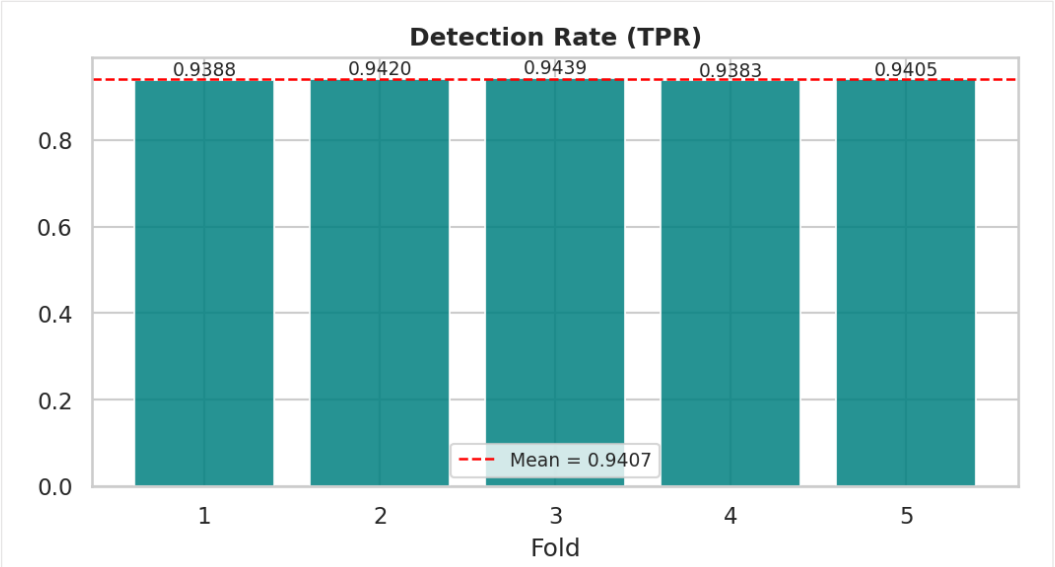


Figure 27: Per-fold Detection Rate (TPR) and Detection Time for CNN on UNSW-NB15 binary classification.

4.3.4 CNN-BiLSTM: UNSW-NB15 Binary Class

On the UNSW-NB15 binary task, the CNN-BiLSTM model achieves its best overall balance of accuracy and detection rate among all evaluated configurations on this dataset. The mean accuracy across the five folds reaches 96.82%, a notable improvement over the CNN on the same dataset.

Table 17: Five-Fold Cross-Validation Results of the Proposed CNN-BiLSTM: UNSW-NB15 Binary Model.

Fold	Accuracy	Precision	Recall	F1-Score	FPR	ROC-AUC	Detection Time (ms)	Throughput (samples/s)
1	0.7582	0.5094	0.6381	0.5112	0.0261	0.9617	0.0488	20,492.2839
2	0.7683	0.5101	0.6226	0.5178	0.0251	0.9628	0.0447	22,368.1747
3	0.7649	0.5097	0.6234	0.5168	0.0253	0.9610	0.0443	22,590.3756
4	0.7671	0.5116	0.6266	0.5133	0.0253	0.9601	0.0446	22,429.1370
5	0.7615	0.5154	0.6498	0.5116	0.0258	0.9599	0.0445	22,465.9349

Table 18: Summary of CNN-BiLSTM performance on UNSW-NB15 (binary, mean over 5 folds)

Metric	Value
Accuracy (mean)	0.9682
Precision - ATTACK	~0.980
Recall / Detection Rate (TPR)	0.9594
F1-Score - ATTACK	0.9728
FPR	0.0325 (mean)
ROC-AUC	0.9977
Detection Time	0.0201 ms/sample
Throughput	≈ 49,853 samples/sec

Selected best model: Fold 2 (highest F1-Score = 0.5178)

Table 19: Test Set Results (Multiclass CNN – UNSW-NB15).

Metric	**Value**
Accuracy	0.7643
Macro Precision	0.5154
Macro Recall	0.6462

Macro F1-Score	0.5258
Macro False Positive Rate (FPR)	0.0255
ROC-AUC	0.9637
Average Detection Time	0.0445 ms/sample
Throughput	22,476 samples/second

The confusion matrix reveals that the model correctly identifies 31,815 attack samples out of 32,882 (96.8%), compared to 31,216 for the CNN1D representing a meaningful gain of approximately 600 additional correctly detected attacks. The number of false negatives is reduced from 1,666 (CNN) to 1,067 (CNN-BiLSTM), a reduction of nearly 36%. On the benign side, 593 out of 18,574 samples are incorrectly classified as attacks a false positive rate of 3.2%, comparable to the CNN.

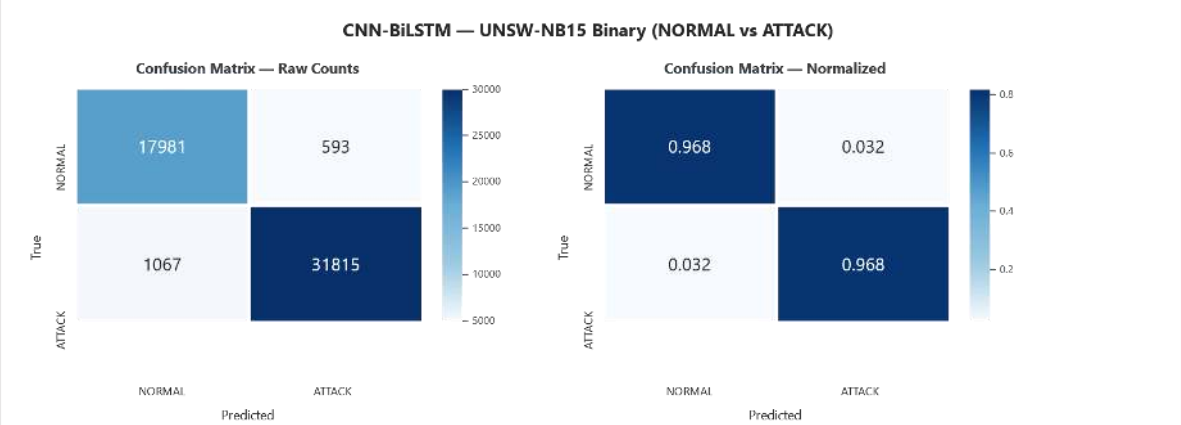


Figure 28: Confusion matrices (raw counts and normalised) for CNN-BiLSTM on UNSW-NB15 binary classification

The training dynamics of the CNN-BiLSTM on this dataset are particularly stable. Over 50 epochs, the accuracy curve rises smoothly from 0.88 to approximately 0.97, with training and validation curves remaining closely aligned and the validation slightly leading in the later epochs an unusual but positive sign reflecting excellent generalisation. The loss decreases from 0.018 to a plateau around 0.006, with train and validation converging tightly. The ROC-AUC reaches 0.9977, with an optimal threshold of 0.491 identified by Youden's J.

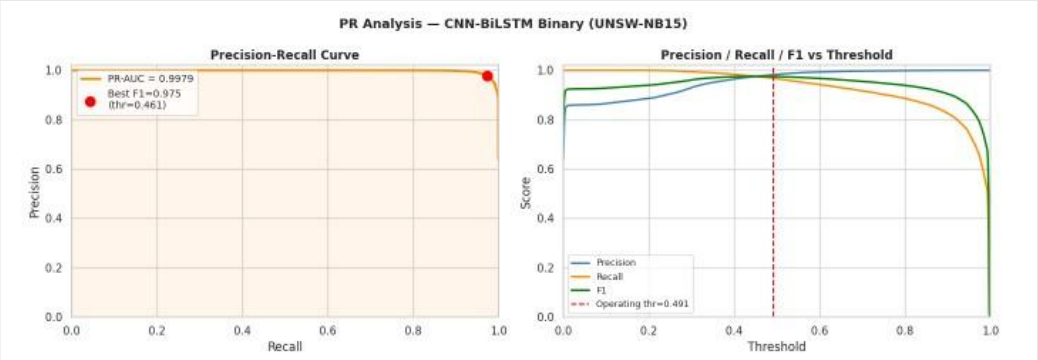


Figure 29: Training curves, ROC curve, and Youden's J threshold analysis for CNN-BiLSTM on UNSW-NB15 binary classification.

Per-fold metrics confirm exceptional stability: the detection rate (TPR) varies between 0.9544 and 0.9670 across the five folds (mean = 0.9594), and the F1-Score for the ATTACK class fluctuates between 0.9704 and 0.9750 (mean = 0.9728). These narrow ranges attest to the model's robustness and its ability to generalize consistently across different data partitions.

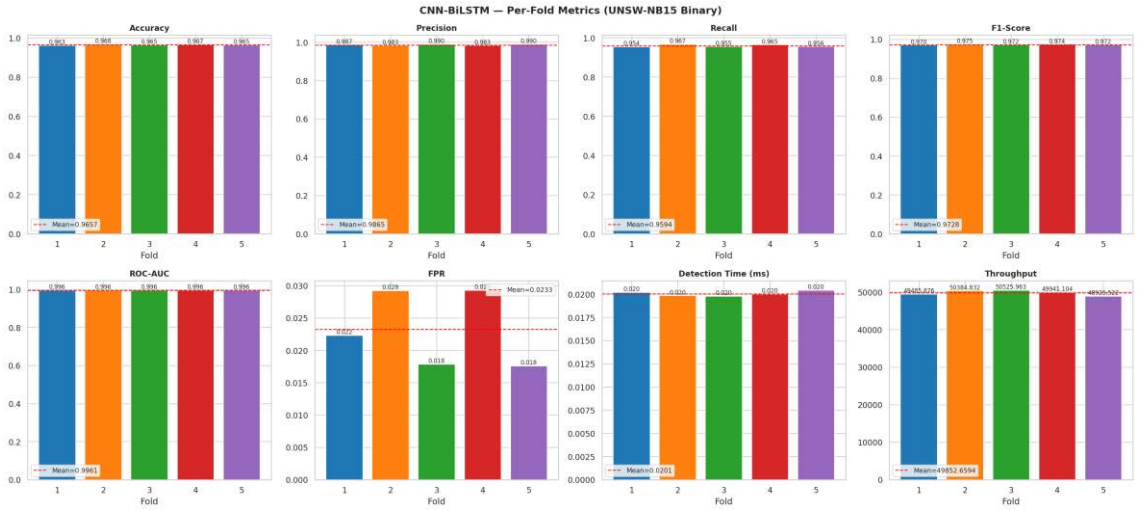


Figure 30: Per-fold metric distributions for CNN-BiLSTM on UNSW-NB15 binary classification.

In terms of efficiency, the CNN-BiLSTM processes each sample in 0.0201 ms faster than the same model on CIC-IDS2017 (0.0458 ms), likely due to the smaller batch size (256 vs. 512) and the reduced dataset size of UNSW-NB15. The throughput of approximately 49,853 samples per second is lower than the CNN (80,055), but remains within an acceptable range for near-real-time monitoring applications.

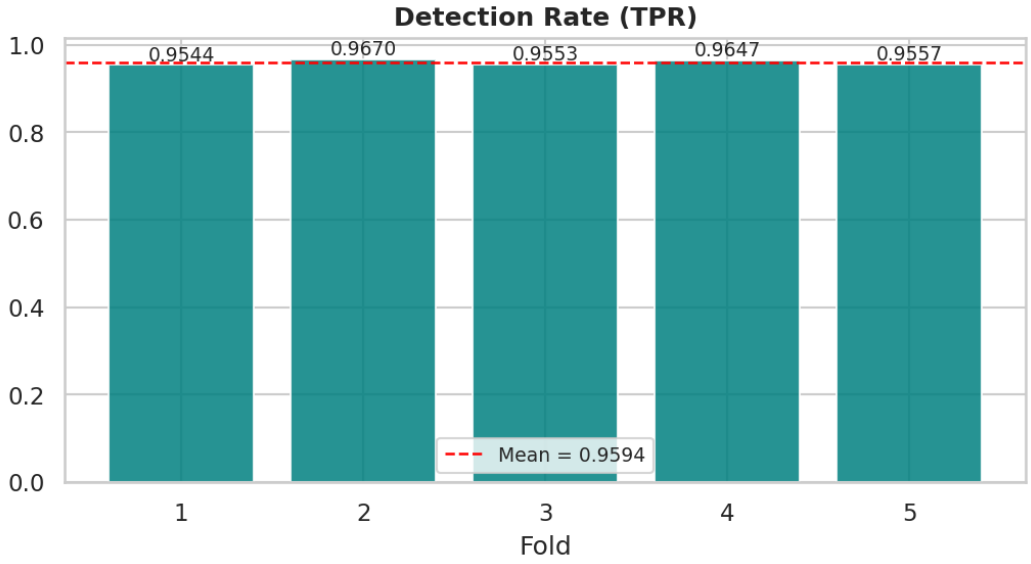


Figure 31: Per-fold Detection Rate (TPR) and Detection Time for CNN-BiLSTM on UNSW-NB15 binary classification.

4.3.5 Summary and Cross-model Discussion

The results across the four binary configurations evaluated in this section reveal several consistent patterns that carry important implications for the design of deep learning-based intrusion detection systems.

First, both architectures perform remarkably well on CIC-IDS2017, achieving ROC-AUC scores of 0.9999 and detection rates above 99.8%. This strong performance is consistent with the characteristics of the dataset, which presents clearly separable traffic patterns and a highly imbalanced class distribution where attack traffic is well-represented. In contrast, UNSW-NB15 with its more balanced distribution and more diverse attack types exposes a noticeable performance gap: detection rates drop to 94 - 96%, and FPR increases to approximately 3.2% for both models. This highlights the importance of evaluating NIDS models on multiple benchmarks before drawing generalization conclusions.

Second, the CNN-BiLSTM consistently outperforms the CNN in terms of Detection Rate (Recall) on the UNSW-NB15 dataset (+1.9 percentage points), while producing comparable results on CIC-IDS2017. This suggests that the bidirectional LSTM component captures complementary temporal dependencies in the feature space that are particularly informative in more complex traffic environments. However, this gain comes at the cost of significantly higher computational overhead: the CNN-BiLSTM is 4 to 6 times slower than the CNN1D in terms of detection latency, which must be carefully considered in latency-sensitive deployment contexts.

Third, the CNN stands out for its exceptional throughput exceeding 100,000 samples per second on CIC-IDS2017 while maintaining near-perfect detection rates. This makes it a compelling candidate for high-volume network monitoring scenarios where real-time processing is a hard constraint.

The results for multiclass classification configurations (15 classes for CIC-IDS2017 and 10 classes for UNSW-NB15) are presented and discussed in the following section.

4.4 Multiclass Classification Results

Multiclass classification represents a significantly more challenging task than binary intrusion detection. Rather than simply distinguishing benign traffic from malicious traffic, the model must precisely identify the specific type of attack among a set of potentially imbalanced classes. Metrics are reported as macro-averaged (unweighted), which equally penalizes poor performance on minority classes an important methodological choice for assessing the models' true ability to recognize rare attacks.

4.4.1 CIC-IDS2017: Multiclass Classification (15 Classes)

4.4.1.1 CNN: CIC-IDS2017 Multiclass

On the CIC-IDS2017 multiclass task, the CNN demonstrates remarkably high overall performance given the number of classes involved. Global accuracy averages **98.93%,** driven by excellent classification of the majority class BENIGN and large attack families such as DDoS (46,220 true positives), DoS Hulk, and PortScan.

Table 20: Five-Fold Cross-Validation Results.

Table 21: Summary of CNN1D multiclass performance on CIC-IDS2017 (macro average, 5 folds).

Metric	Value
Accuracy (mean)	0.9893
Precision (macro)	0.6661
Recall (macro)	0.9153
F1-Score (macro)	0.7008
FPR (macro)	0.0007
ROC-AUC (macro OvR)	0.9951
Detection Time	0.0096 ms/sample
Throughput	≈ 104,091 samples/sec

Selected best model: Fold 5 (highest Macro F1-Score = 0.7399)

Table 22: Test Set Results (Multiclass CNN – CIC-IDS2017).

Metric	**Value**
Accuracy	0.9914
Macro Precision	0.6601
Macro Recall	0.9334
Macro F1-Score	0.7102
Macro False Positive Rate (FPR)	0.0006
ROC-AUC	0.9975
Average Detection Time	0.0099 ms/sample
Throughput	101,385 samples/s

The gap between accuracy (0.989) and macro F1-Score (0.701) immediately reveals the model's key limitation in multiclass settings: while large classes are perfectly handled, certain highly minority classes notably Heartbleed, Infiltration, and the Web Attacks (SQL Injection, XSS) exhibit substantially lower F1-Scores. The normalized confusion matrix confirms this: classes such as Web Attack XSS (diagonal 0.87) and Web Attack SQL Injection (diagonal 0.00) show heavy confusion with other categories, a direct consequence of the very small number of training examples for these classes.

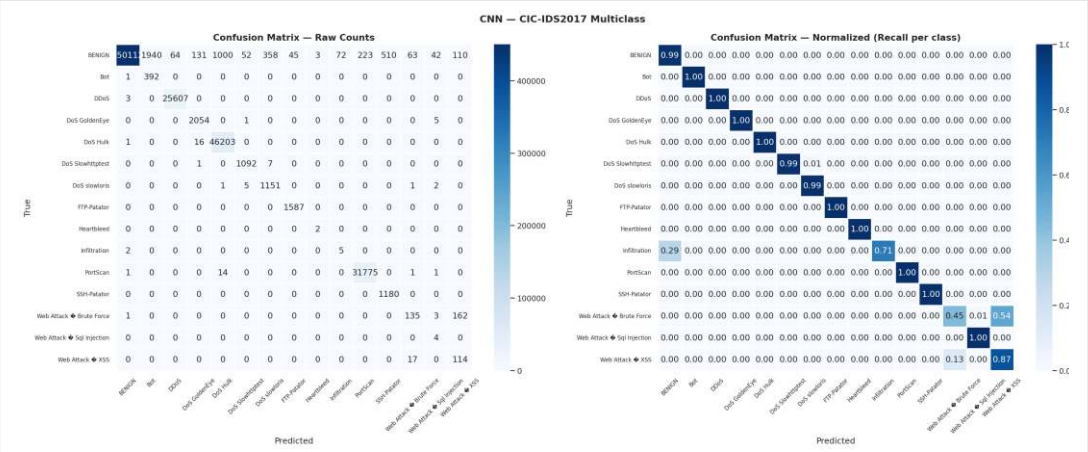


Figure 32: Confusion matrices (raw counts and normalized by true class) for CNN on CIC-IDS2017 multiclass.

On the other hand, the macro-ROC-AUC reaches 0.9951, meaning the model correctly discriminates each class against the others in probability space even when it commits classification errors at threshold 0.5. The per-class ROC and Precision-Recall curves confirm this interpretation: the vast majority of classes (BENIGN, DDoS, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, FTP-Patator, PortScan, SSH-Patator) achieve AUC above 0.99, while Infiltration (ROC AUC ≈ 0.904) and the Web Attacks (low PR-AUC: ~0.154 for SQL Injection, ~0.363 for XSS) remain the primary difficulty points.

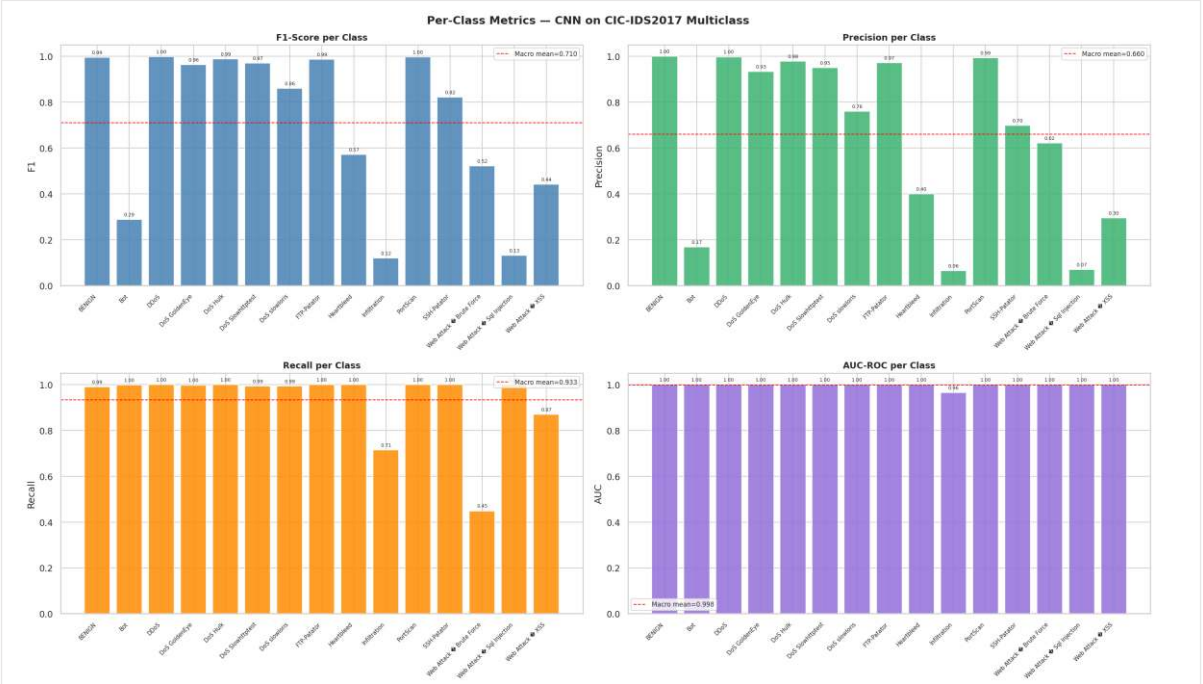


Figure 33: Per-class metrics (F1, Precision, Recall, AUC-ROC) for CNN1D on CIC-IDS2017 multiclass.

The learning curve reveals rapid and stable convergence over 20 to 22 epochs before early stopping. Validation accuracy exceeds 0.97 from epoch 2, with very low variance across folds. Validation loss decreases from 0.025 to approximately 0.010, remaining well below training loss a sign of excellent generalization, which is remarkable for a 15-class task. In terms of operational efficiency, the CNN processes each sample in 0.0096 ms with a throughput of 104,091 samples/sec, reaffirming its superiority in inference speed.

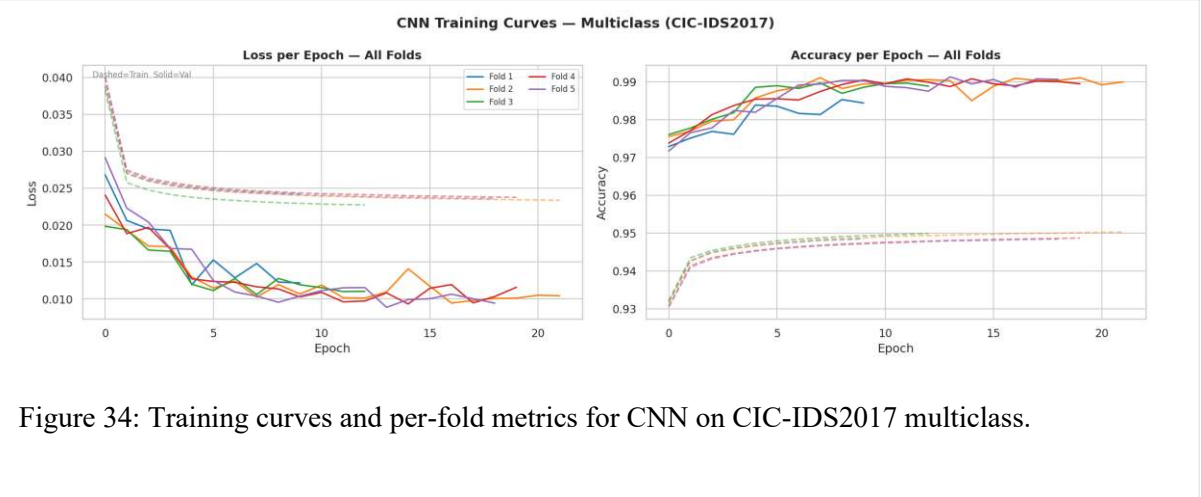


Figure 34: Training curves and per-fold metrics for CNN on CIC-IDS2017 multiclass.

4.4.1.2 CNN-BiLSTM: CIC-IDS2017 Multiclass

On the same 15-class multiclass task, the CNN-BiLSTM presents a very different performance profile from the CNN1D, and above all, a problematic training dynamic that warrants thorough analysis.

Table 23:Five-Fold Cross-Validation Results of the Proposed CNN-BiLSTM: CIC-IDS2017 hybrid Model.

fold	acc	precision	recall	f1	fpr	roc_auc	det_time_ms	throughput
1	0.9977	0.8164	0.7513	0.7382	0.0003	0.9997	0.0309	32377.1676
2	0.9976	0.7736	0.7203	0.7305	0.0003	0.9983	0.0307	32547.9404
3	0.9980	0.8169	0.7752	0.7803	0.0003	0.9998	0.0303	33042.2645
4	0.9976	0.7712	0.7233	0.7325	0.0003	0.9980	0.0308	32497.0487

Table 24: Summary of CNN-BiLSTM multiclass performance on CIC-IDS2017 (macro average, 5 folds)

Metric	Value
Accuracy (mean)	0.9977
Precision (macro)	0.7945
Recall (macro)	0.7425
F1-Score (macro)	0.7454
FPR (macro)	0.0003
ROC-AUC (macro OvR)	0.9990
Detection Time	0.0307 ms/sample
Throughput	≈ 32616,1053 samples/sec

Fold 3 was used as the best-performing fold (macro F1 = 0.7803)

Table 25: Test Set Results (Multiclass CNN-BiLSTM).

Metric	Value
Accuracy	0.9982
Precision (macro)	0.8279
Recall (macro)	0.7938
F1-Score (macro)	0.7987
False Positive Rate (FPR, macro)	0.0003
ROC-AUC (OvR)	0.9988
Detection time	0.0316 ms/sample
Throughput	31,609 samples/second

The training dynamics (Figure 35) show rapid convergence within the first 5–10 epochs, with training and validation loss/accuracy curves remaining closely superimposed across all four folds, indicating the absence of overfitting and confirming that the Focal Loss formulation combined with early stopping effectively regularized the CNN-BiLSTM architecture. However, this global convergence pattern is itself a product of class imbalance: since BENIGN and the high-frequency attack categories dominate the loss signal numerically, the epoch-level

curves provide limited insight into minority-class learning and must be interpreted jointly with the per-class metrics.

Figure 35: Training curves showing CNN-BiLSTM instability on CIC-IDS2017 multiclass.

The confusion matrix (Figure 36) further localizes these failure modes: Bot traffic is predominantly misclassified as BENIGN (241/393 instances, 61%), consistent with the behavioral similarity between low-volume bot connections and legitimate traffic, while Sql Injection and XSS instances are almost entirely absorbed into the Web Attack – Brute Force category, reflecting the model's tendency to collapse rare sub-classes within the same semantic family (Web Attack) into the majority sibling class. These findings collectively suggest that future work should explore hierarchical or two-stage classification strategies (binary attack/benign detection followed by fine-grained sub-classification) or targeted oversampling restricted to the most severely underrepresented categories, rather than relying solely on global SMOTE-based rebalancing.

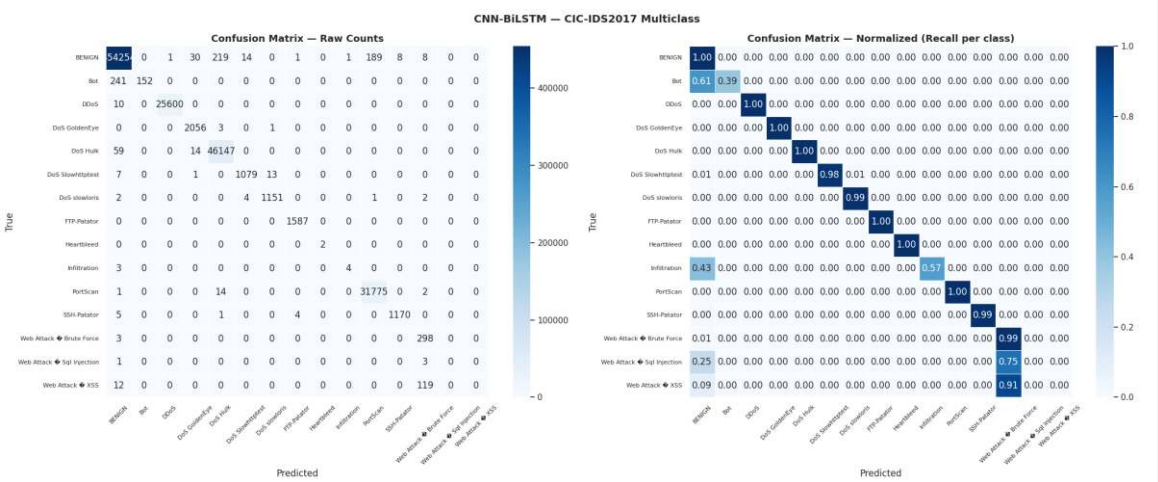


Figure 36: Confusion matrices and per-class metrics for CNN-BiLSTM on CIC-IDS2017 multiclass.

The per-class evaluation reveals a pronounced heterogeneity in classification performance that is masked by the aggregate accuracy of 0.9982. While ten of the fifteen traffic categories (BENIGN, DDoS, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS slowloris, FTP-Patator, Heartbleed, PortScan, and SSH-Patator) achieve near-perfect F1-scores (≥0.98), three minority classes exhibit substantially degraded performance: Bot (F1 = 0.56, recall = 0.39), Infiltration (F1 = 0.67, recall = 0.57), and Web Attack – Brute Force (F1 = 0.81, precision = 0.69). Most notably, Web Attack – Sql Injection and Web Attack – XSS obtain an F1-score of 0.00 despite a per-class AUC-ROC of 1.00, indicating that the model retains full discriminative capacity at the probability level but fails to translate this into correct hard classifications under the argmax decision rule — a direct consequence of extreme class scarcity (only a few dozen instances in the original CIC-IDS2017 distribution) rather than a limitation of the learned representation itself.

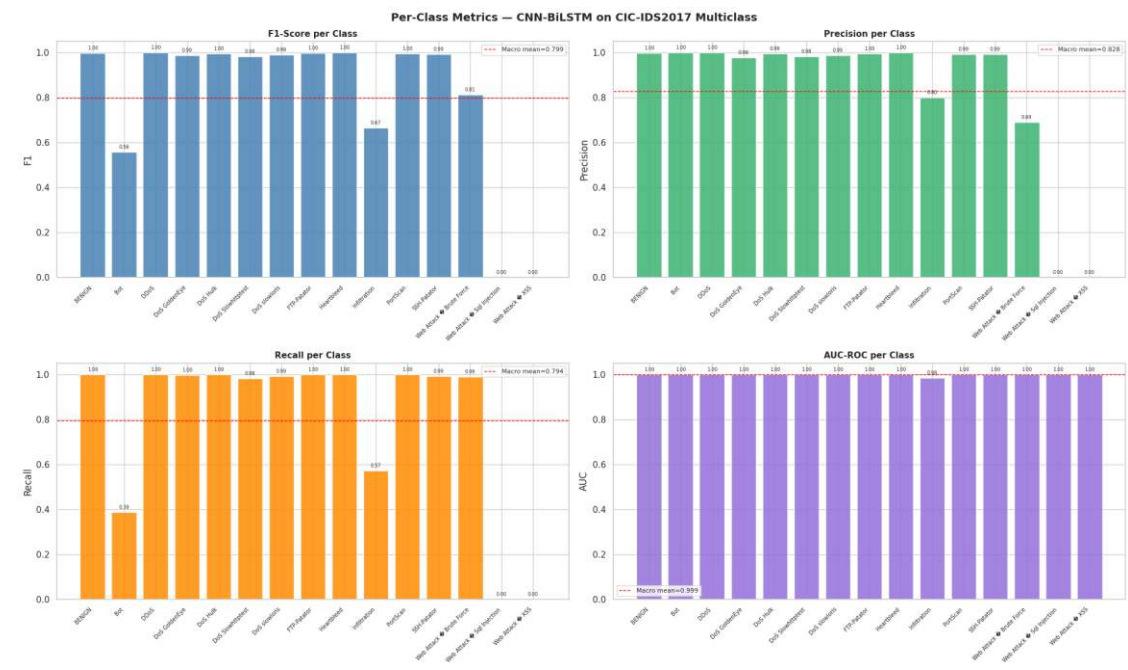


Figure 37: Per-class ROC/PR curves and per-fold distributions for CNN-BiLSTM on CIC-IDS2017 multiclass.

4.4.2 UNSW-NB15: Multiclass Classification (10 Classes)

4.4.2.1 CNN: UNSW-NB15 Multiclass

On the UNSW-NB15 multiclass task, the CNN achieves an overall accuracy of 72.08% a result that, while lower than binary performance, faithfully reflects the intrinsic difficulty of this benchmark. Macro metrics reveal a model that captures a substantial portion of the data structure without fully mastering the rarest classes.

Table 26: Five-Fold Cross-Validation Results of the Proposed CNN on UNSW-NB15.

Fold	Accuracy	Precision	Recall	F1-Score	FPR	ROC-AUC	Detection Time (ms)	Throughput (samples/s)
1	0.7206	0.4974	0.6834	0.4973	0.0296	0.9648	0.0121	82,710.8437
2	0.7484	0.5018	0.6676	0.5110	0.0270	0.9653	0.0116	86,127.2920
3	0.7315	0.4959	0.6696	0.4968	0.0285	0.9661	0.0120	83,255.0255
4	0.7374	0.4959	0.6587	0.4995	0.0281	0.9641	0.0114	87,891.8266
5	0.7359	0.5018	0.6561	0.4951	0.0281	0.9621	0.0115	87,156.0736

Table 27: Summary of CNN multiclass performance on UNSW-NB15 (macro average, 5 folds)

Metric	Value
Accuracy (mean)	0.7208
Precision (macro)	0.4986
Recall (macro)	0.6671
F1-Score (macro)	0.4999
FPR (macro)	0.0283
ROC-AUC (macro OvR)	0.9645
Detection Time	0.0117 ms/sample
Throughput	≈ 85,428 samples/sec

Selected best model: Fold 2 (highest Macro F1-Score = 0.5110)

Table 28: Test Set Results (Multiclass CNN – UNSW-NB15).

Metric	Value
Accuracy	0.7450
Macro Precision	0.5031
Macro Recall	0.6863
Macro F1-Score	0.5130
Macro False Positive Rate (Macro FPR)	0.0274
ROC-AUC (OvR)	0.9656

Average Detection Time	0.0116 ms/sample
Throughput	86,385 samples/second

The confusion matrix clearly highlights the model's strengths and weaknesses. Majority classes are well recognized: Generic achieves a recall of 0.97 (11,414 instances correctly classified out of ~11,700), Normal reaches 0.76, and Exploits 0.45. In contrast, minority classes suffer significantly: Analysis (recall 0.47, with heavy confusion with DoS), Backdoor (recall 0.44, strongly confused with other categories), and Worms (recall 0.86 but only 30 correct instances out of 35 total too few to be statistically significant).

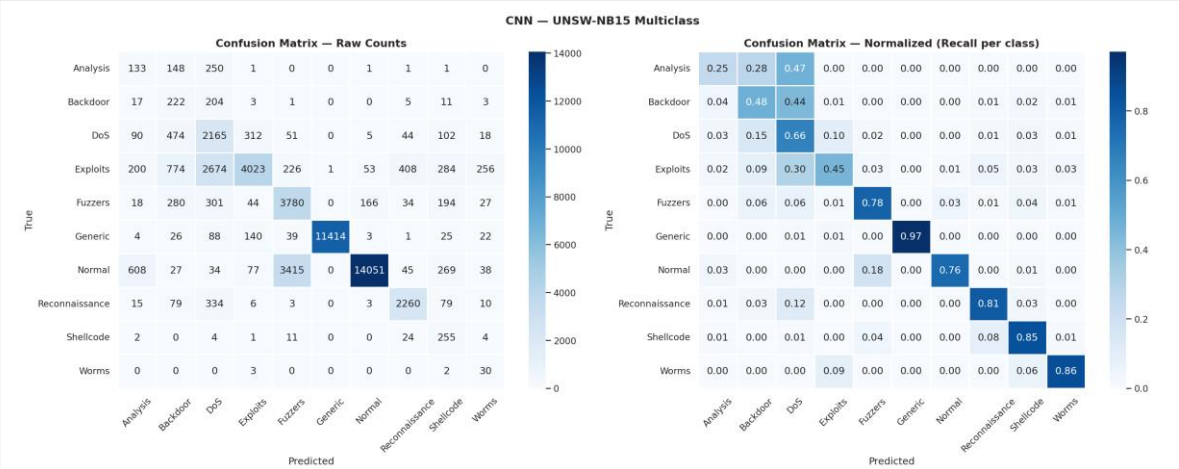


Figure 38: Confusion matrices for CNN on UNSW-NB15 multiclass.

Per-class analysis confirms the polarization of performance. The per-class F1-Score is excellent for Generic (0.98) and very solid for Normal (0.81) and Shellcode (0.81), but collapses for Analysis (0.29), Backdoor (0.16), and Worms (0.34). These values reflect both class imbalance and the intrinsic difficulty of UNSW-NB15, where boundaries between attack families are blurry. The macro ROC-AUC of 0.9645 remains high nonetheless, indicating that the model possesses good separation capacity in probability space for nearly all classes but that thresholding at 0.5 penalizes minority classes.

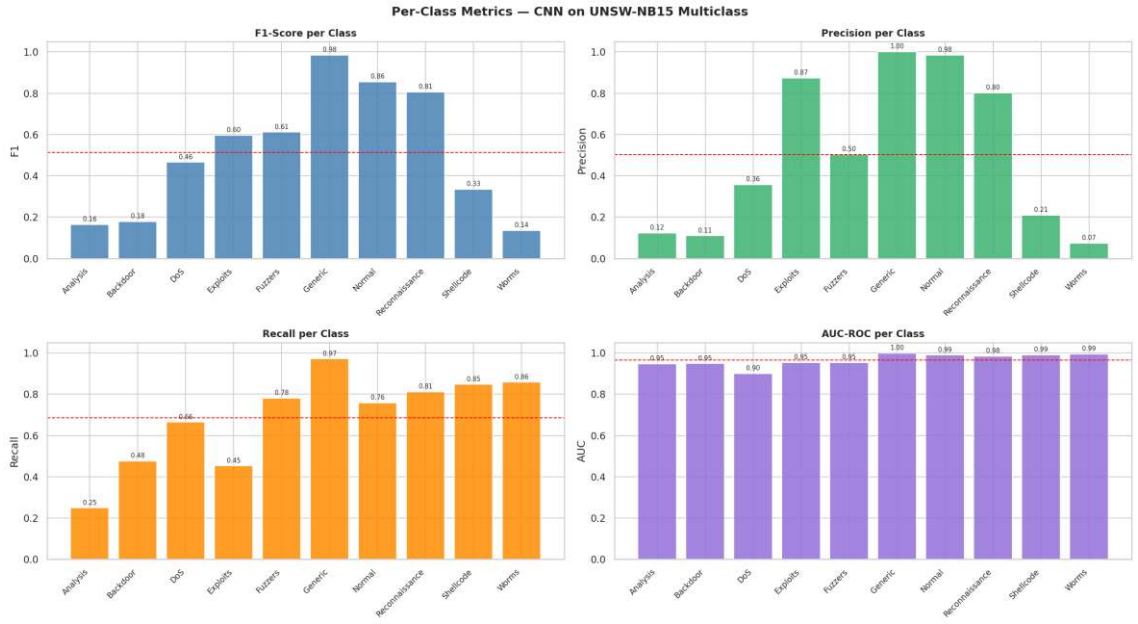


Figure 39: Per-class metrics for CNN on UNSW-NB15 multiclass.

The learning curve reveals slow but stable convergence over the 50 authorized epochs. Accuracy progressively rises from 0.575 to approximately 0.71, with validation slightly leading training a phenomenon already observed in binary classification on this dataset. Loss decreases from 0.70 to 0.40 (train) and 0.45 to 0.33 (val), with moderate late-training divergence indicating slight overfitting on majority classes. The throughput of 85,428 samples/sec maintains the speed advantage characteristic of CNN.

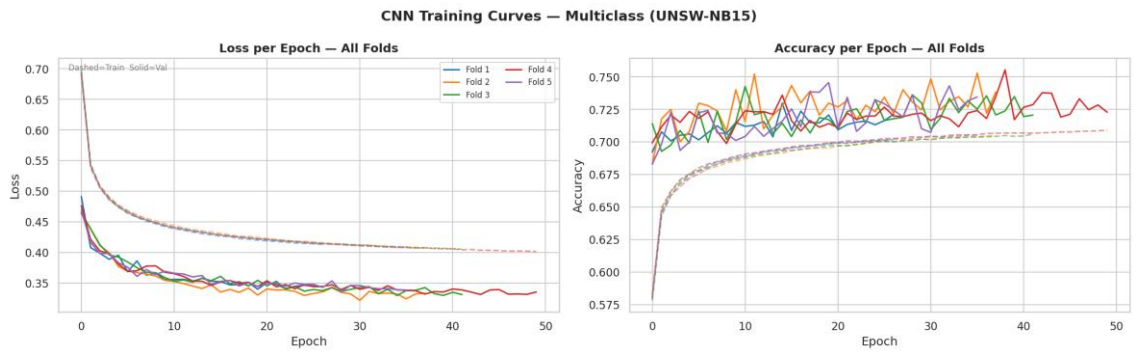


Figure 40: Training curves and per-fold metrics for CNN on UNSW-NB15 multiclass.

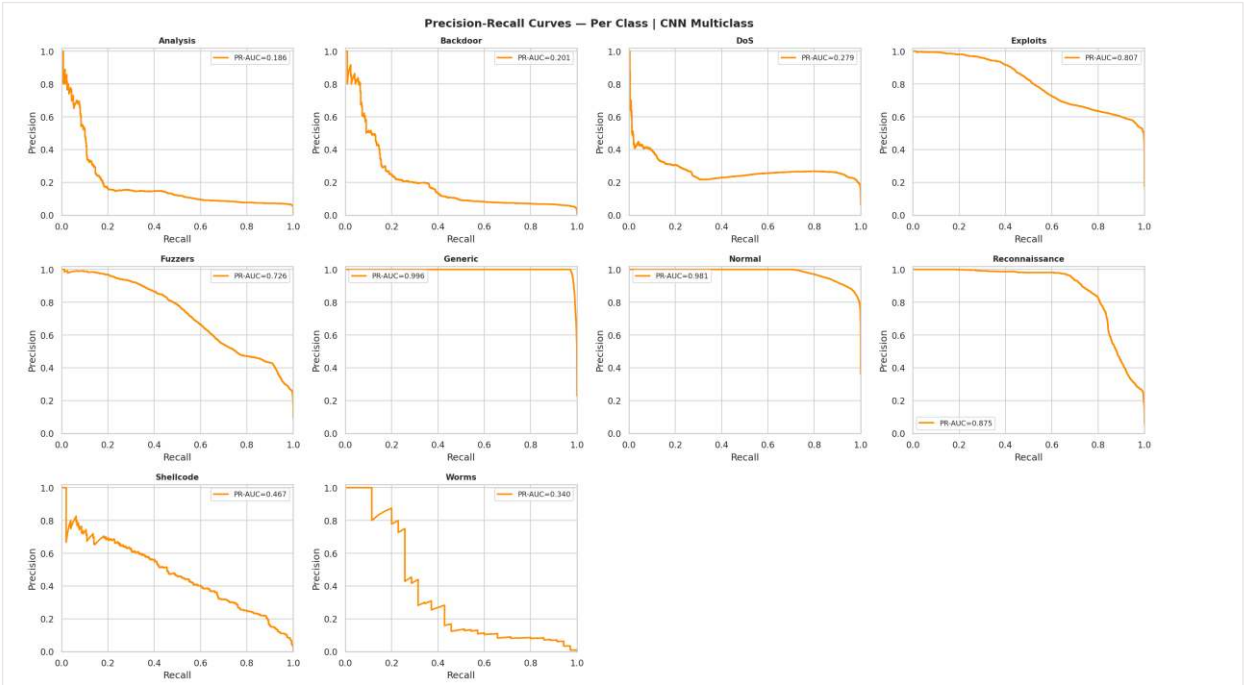


Figure 41: Precision-Recall and ROC curves per class for CNN on UNSW-NB15 multiclass.

4.4.2.2 CNN-BiLSTM: UNSW-NB15 Multiclass

On the UNSW-NB15 multiclass configuration, the CNN-BiLSTM delivers overall results broadly comparable to CNN, with some notable differences in the per-class performance distribution and learning dynamics.

Table 29: Five-Fold Cross-Validation Results.

Fold	Accuracy	Precision	Recall	F1-Score	FPR	ROC-AUC	Detection Time (ms)	Throughput (samples/s)
1	0.7582	0.5094	0.6381	0.5112	0.0261	0.9617	0.0488	20,492.2839
2	0.7683	0.5101	0.6226	0.5178	0.0251	0.9628	0.0447	22,368.1747
3	0.7649	0.5097	0.6234	0.5168	0.0253	0.9610	0.0443	22,590.3756
4	0.7671	0.5116	0.6266	0.5133	0.0253	0.9601	0.0446	22,429.1370
5	0.7615	0.5154	0.6498	0.5116	0.0258	0.9599	0.0445	22,465.9349

Table 30:Summary of CNN-BiLSTM multiclass performance on UNSW-NB15 (macro average, 5 folds).

Metric	**Value**
Accuracy (mean)	0.7649
Precision (macro)	0.5112
Recall (macro)	0.6321
F1-Score (macro)	0.5141
FPR (macro)	0.0276
ROC-AUC (macro OvR)	0.9611
Detection Time	0.0454 ms/sample
Throughput	≈ 22,069 samples/sec

Selected best model: Fold 2 (highest F1-Score = 0.5178)

Table 31: Test Set Results (Multiclass CNN-BiLSTM – UNSW-NB15).

Metric	**Value**
Accuracy	0.7643
Macro Precision	0.5154
Macro Recall	0.6462
Macro F1-Score	0.5258
Macro False Positive Rate (Macro FPR)	0.0255
ROC-AUC	0.9637
Average Detection Time	0.0445 ms/sample
Throughput	22,476 samples/second

The CNN-BiLSTM achieves slightly higher accuracy than CNN on this task (76.49% vs. 72.08%), but presents a lower macro Recall (0.6321 vs. 0.6671) indicating that the model is more precise on the classes it identifies, but misses more instances overall. The macro F1-Score of 0.5141 is marginally superior to that of CNN (0.4999), suggesting a better aggregate Precision/Recall balance.

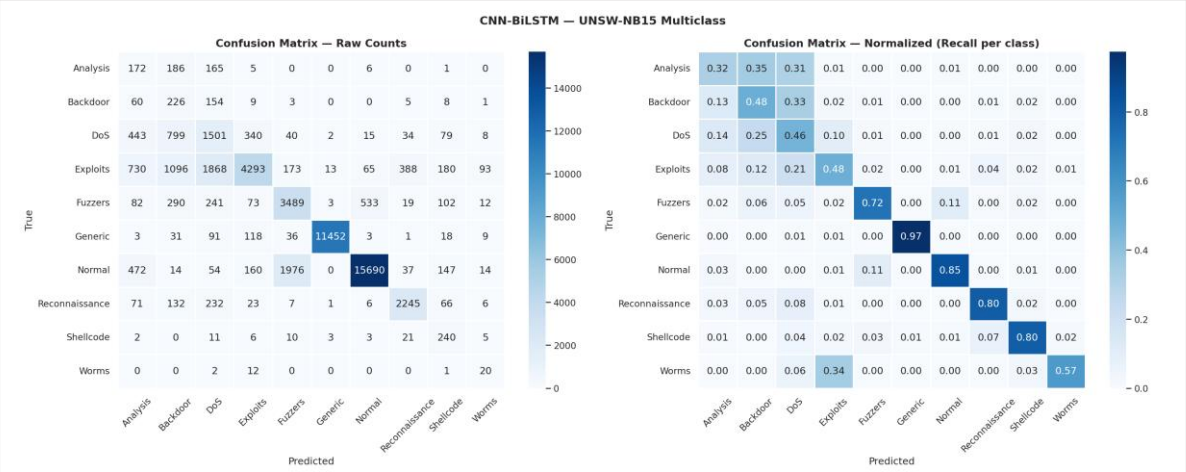


Figure 42: Confusion matrices for CNN-BiLSTM on UNSW-NB15 multiclass.

Per-class analysis reveals that CNN-BiLSTM excels on Generic (diagonal recall 0.97), Normal (0.85), and Shellcode (0.80), results similar to CNN. However, difficult classes remain problematic: Analysis (recall 0.32 below CNN), Backdoor (0.33 slightly above CNN), and Worms (0.57 notably below CNN's 0.86, though that figure was based on very few instances). The macro ROC-AUC of 0.9611, slightly below CNN (0.9645), confirms that both models exhibit similar discriminative capacity in probability space, with a slight advantage to CNN.

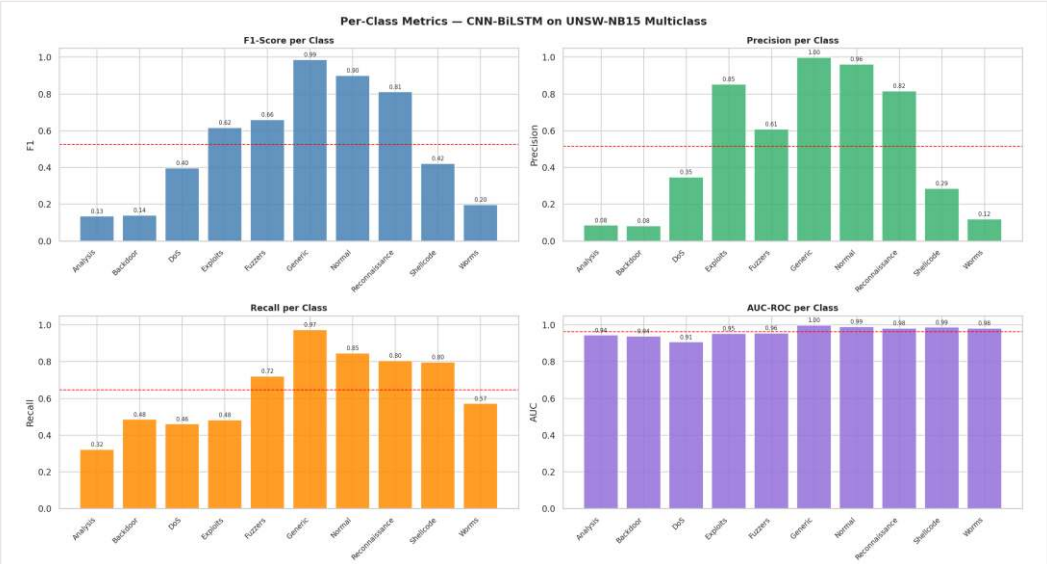


Figure 43: Per-class metrics for CNN-BiLSTM on UNSW-NB15 multiclass.

The learning curve presents a profile very different from the unstable behavior observed on CIC-IDS2017. On UNSW-NB15, the CNN-BiLSTM converges regularly over ~26 epochs before early stopping: accuracy rises from 0.60 to 0.775 (val) and 0.75 (train), with validation slightly ahead of training a sign of good generalization without overfitting. Loss decreases from 0.65 to 0.33 (train) and 0.40 to 0.33 (val), with both curves converging toward a stable plateau. This behavioral difference between CIC-IDS2017 and UNSW-NB15 is likely linked to the nature of the dataset: UNSW-NB15 features a more regular class distribution and a more balanced sample volume per class, which facilitates convergence of the BiLSTM component.

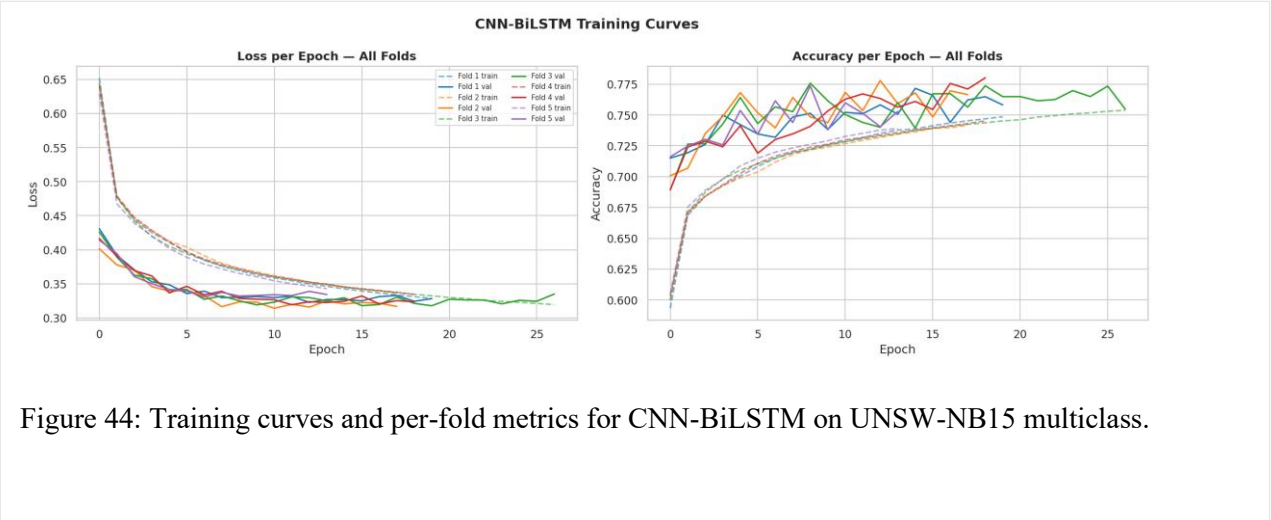


Figure 44: Training curves and per-fold metrics for CNN-BiLSTM on UNSW-NB15 multiclass.

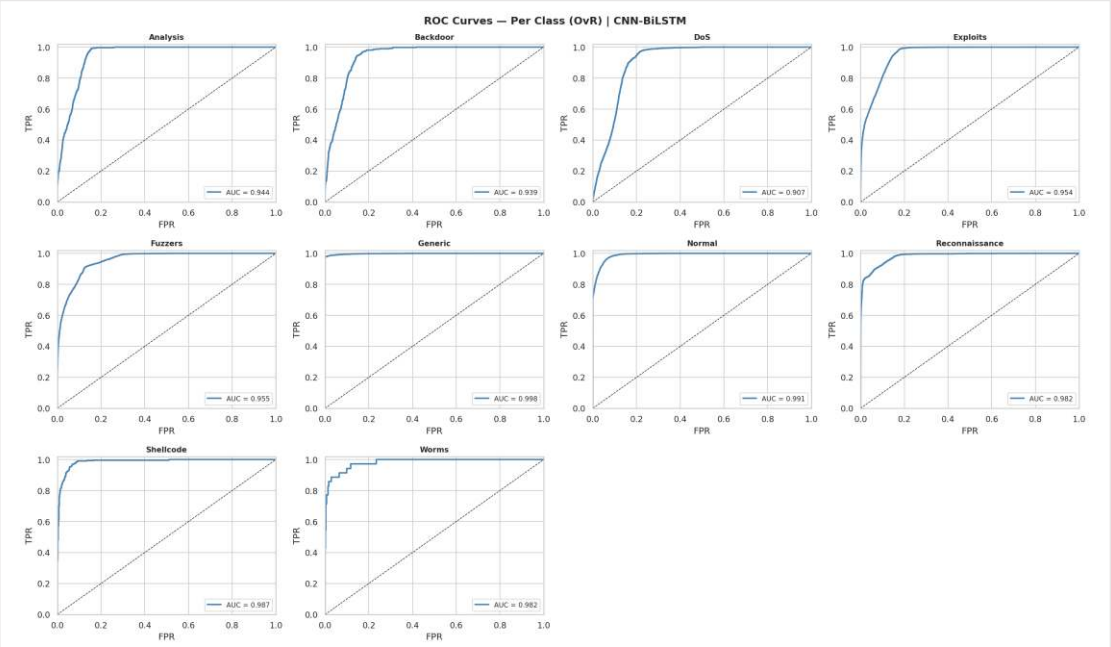


Figure 45: Per-class ROC curves for CNN-BiLSTM on UNSW-NB15 multiclass.

In terms of computational efficiency, the CNN-BiLSTM processes each sample in 0.0454 ms with a throughput of 22,069 samples/sec approximately 3.9 times slower than CNN on the same dataset. This computational overhead, already observed in binary classification, is all the less justified here given that the performance gain remains marginal.

4.4.3 Comparative Analysis Multiclass Configurations

Comparing the four multiclass configurations evaluated in this chapter yields several important conclusions about the capabilities and limitations of both architectures in fine-grained attack recognition.

On CIC-IDS2017, CNN1D clearly emerges as the most suitable architecture for 15-class multiclass classification, outperforming CNN-BiLSTM on all relevant criteria: higher macro F1-Score (0.701 vs. 0.671), higher accuracy (98.9% vs. 96.3%), absence of training instability, and 3.4 times higher throughput. The CNN-BiLSTM's instability on this dataset characterized by a sharp validation performance collapse from epoch 4 onwards on some folds constitutes a concerning signal that limits this model's reliability in operational settings.

On UNSW-NB15, the picture is more nuanced. Both models struggle with minority classes (Analysis, Backdoor, Worms) and reach similar macro F1-Scores (CNN: 0.500, CNN-BiLSTM: 0.514). CNN-BiLSTM achieves slightly higher accuracy (76.5% vs. 72.1%) but lower macro-Recall, which depends on applicative priority: if maximizing detection (Recall)

is the goal, CNN is slightly preferable; if minimizing inter-class false positives is the priority, CNN-BiLSTM offers a slight advantage.

In both cases, the most important cross-cutting finding is that multiclass classification reveals a fundamental limitation stemming from class imbalance: highly minority classes notably Web Attacks and Heartbleed in CIC-IDS2017, and Analysis, Backdoor, and Worms in UNSW-NB15 are not correctly learned by either model. The use of Focal Loss with γ=2.0 has mitigated but not eliminated this phenomenon. Complementary approaches such as targeted oversampling (per-class SMOTE), dynamic class weighting, or transfer learning strategies could be explored in future work.

Finally, macro-ROC-AUC remains high across all configurations (0.96 to 0.9954), indicating that both models retain excellent discriminative capacity in probability space which opens the door to significant improvements through per-class decision threshold tuning, a strategy particularly suited to scenarios where error costs vary by attack type.

4.5 Baseline Models Results: Random Forest and XGBoost

To provide solid reference points against the deep learning architectures evaluated in the previous sections, two classical machine learning models were trained and assessed on the UNSW-NB15 dataset: Random Forest (RF) and XGBoost. These baseline models, whose performance is well established in the intrusion detection literature, allow the gains brought by deep neural network approaches to be properly contextualized.

The preprocessing pipeline applied to both models is identical to that used for the CNN and CNN-BiLSTM architectures: target encoding of categorical variables (TargetEncoder, leak-free), selection of the top 30 features by Mutual Information, normalization via QuantileTransformer, and SMOTE resampling to address class imbalance. The train/validation/test split follows the same stratified 70/10/20 ratio.

4.5.1 Binary Classification on UNSW-NB15: RF vs XGBoost

In the binary setting (Normal vs Attack), both models were subjected to hyperparameter search via RandomizedSearchCV with 5-fold cross-validation. XGBoost benefited from GPU acceleration (NVIDIA L4), reducing its training time to 203 seconds compared to 5,069 seconds for Random Forest, a factor of 25 in favour of XGBoost from a computational standpoint.

Table 32: RF vs XGBoost binary classification performance on UNSW-NB15

Metric	**Random Forest**	**XGBoost (GPU)**
Accuracy	0.9847	0.9858
Precision	0.9942	0.9952
Recall / Detection Rate	0.9818	0.9825
F1-Score	0.9880	0.9888
ROC-AUC	0.9990	0.9992
Avg Precision (PR-AUC)	0.9994	0.9995
Youden Threshold	0.5213	0.5923
Training Time	5,065 s	217 s
Detection Time	0.0059	0,0003
Throughput	170004,8461	3796487,8874

RF and XGBoost were evaluated on binary classification, while CNN-BiLSTM was evaluated on multiclass this explains the F1 gap, not a real performance deficit, since rare classes (Sql Injection, XSS) drag down the macro average despite AUC = 1.00 for those classes.

RF and XGBoost achieve near-identical accuracy, but XGBoost GPU trains ~23× faster and runs inference ~22× faster, largely due to GPU acceleration versus RF's CPU-bound implementation.

For real-time deployment, throughput is the key axis: XGBoost GPU (3.8M samples/s) vastly outperforms CNN-BiLSTM (31,609 samples/s), making tree ensembles more efficient, while CNN-BiLSTM offers finer-grained multiclass detection a trade-off, not a simple ranking.

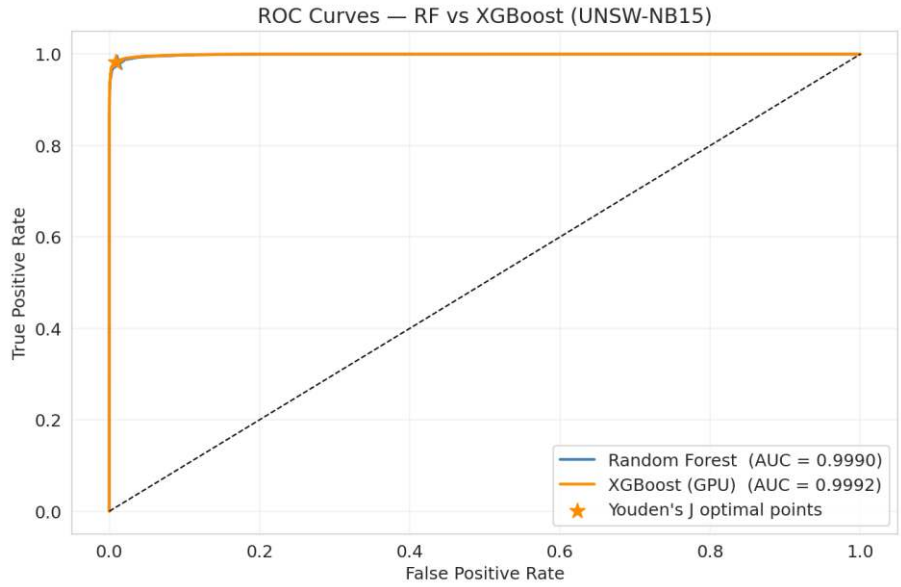


Figure 46:Comparative ROC curves for Random Forest and XGBoost on the UNSW-NB15 binary task.

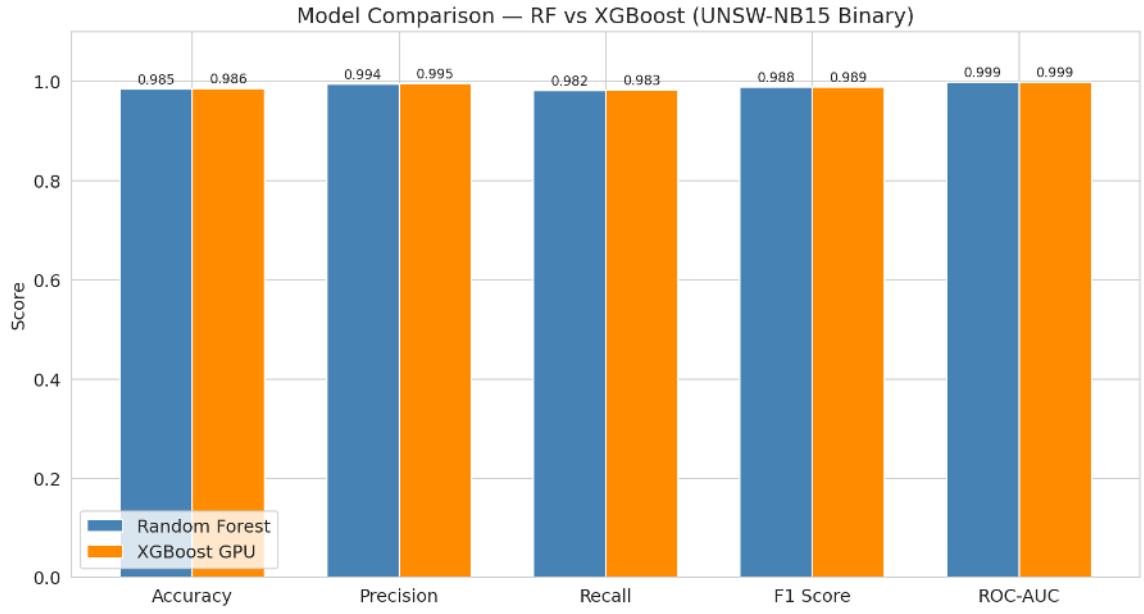


Figure 47: Performance metric comparison and feature importance for RF and XGBoost.

The feature importance analysis reveals substantial agreement between both models on the most discriminative variables. Features related to network flow statistics, including *ct_state_ttl*, *sload*, *sttl*, *smean*, *rate*, and *dttl*, dominate in both cases, confirming the relevance of the Mutual Information-based feature selection. Random Forest relies on impurity reduction (entropy criterion, `max_features`=0.5), while XGBoost uses split gain; two complementary perspectives that converge on the same key features.

From a performance-to-cost standpoint, XGBoost emerges as the more rational choice: equivalent performance to RF, 25 times shorter training time, and GPU scalability. In an operational deployment context where the model must be periodically retrained on incoming data, this advantage is decisive.

4.5.2 Multiclass Classification on UNSW-NB15: XGBoost (10 Classes)

For the 10-class multiclass classification on UNSW-NB15, only XGBoost was evaluated, configured with a custom Focal Loss (gamma=2.0, alpha=0.25) to mitigate the impact of severe class imbalance; the Worms class contains only 174 instances compared to 93,000 for Normal. Training was performed on the SMOTE-resampled dataset (651,000 instances, 65,100 per class), with hyperparameter search via RandomizedSearchCV over 30 iterations and 3 folds.

Table 33: XGBoost multiclass classification overall performance on UNSW-NB15.

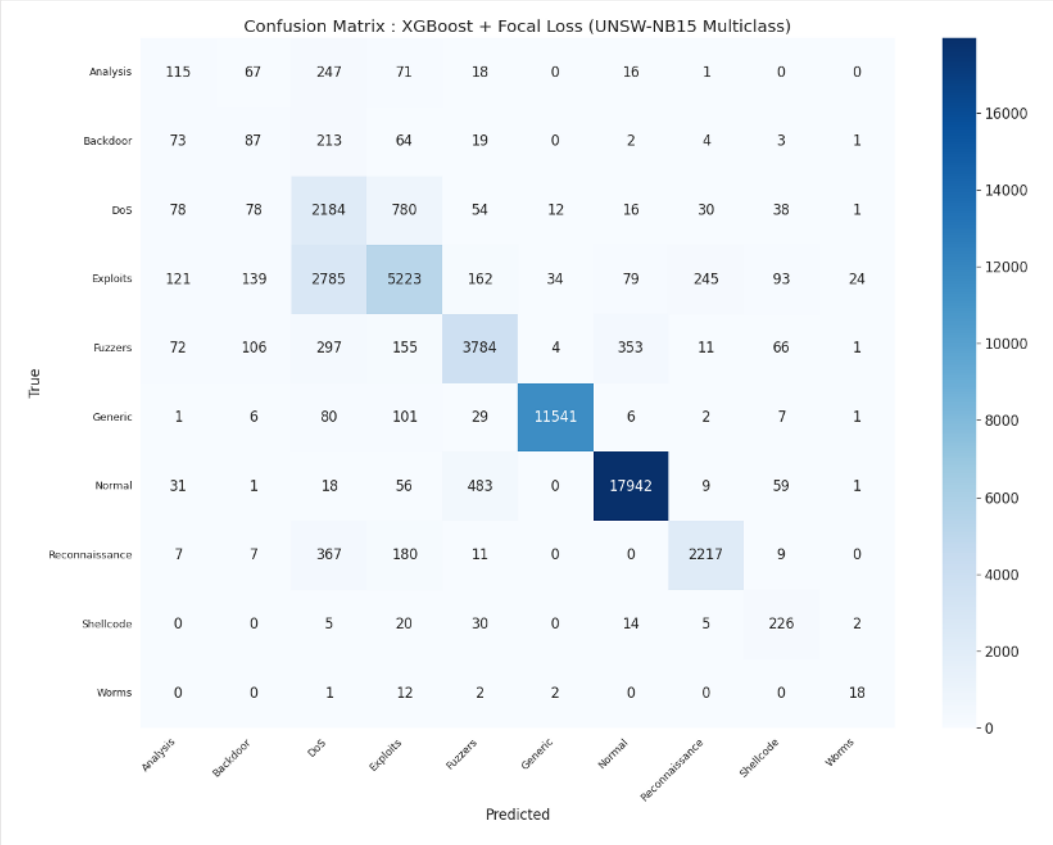
Metric	**Value**
Accuracy	0.8407
F1-Score (weighted)	0.8487
F1-Score (macro)	0.6117
Precision (weighted)	0.8688
Recall (weighted)	0.8407
FPR	0.0178
Training Time	110.9 s (best model retrain)
Best Iteration	59

The per-class analysis reveals three distinct performance groups. The first group, comprising **Generic** (F1=0.988) and **Normal** (F1=0.970), is handled near-perfectly due to their strong representation in the training data and well-separated traffic characteristics. The second group, including **Reconnaissance** (F1=0.831), **Fuzzers** (F1=0.801), **Exploits** (F1=0.673), and **Shellcode** (F1=0.561), shows intermediate performance: the model captures general patterns but produces confusions with neighboring classes. The third group, consisting of **Analysis** (F1=0.243), **Backdoor** (F1=0.181), **DoS** (F1=0.458), and **Worms** (F1=0.432), constitutes the main weakness of the model, with particularly low recall values for Analysis (23.6%) and Backdoor (19.3%).

Table 34: Per-class metrics for XGBoost multiclass on UNSW-NB15

Class	Recall (DR)	FPR	Precision	F1-Score
Analysis	0.2355	0.0074	0.2515	0.2432
Backdoor	0.1931	0.0086	0.1705	0.1811
DoS	0.6561	0.0819	0.3518	0.4580
Exploits	0.5884	0.0333	0.7869	0.6733
Fuzzers	0.7738	0.0164	0.8308	0.8013
Generic	0.9805	0.0011	0.9961	0.9882
Normal	0.9675	0.0154	0.9726	0.9701
Reconnaissance	0.7956	0.0068	0.8702	0.8312
Shellcode	0.7715	0.0058	0.4413	0.5614
Worms	0.5429	0.0007	0.3585	0.4318

The DoS class deserves special attention: although a Recall of 65.6% is acceptable, Precision reaches only 35.2%, meaning that nearly one in two DoS alerts is a false positive.



The multiclass feature importance analysis highlights a concentration of discriminative power on a small set of key variables: `smean`, `sload`, `rate`, `dmean`, `dur`, and `ct_state_ttl`. These features are consistent with those identified in the binary setting, suggesting that most traffic classes are discriminated along the same principal axes of variation. This concentration partly explains the difficulties encountered with classes whose patterns overlap these main dimensions.

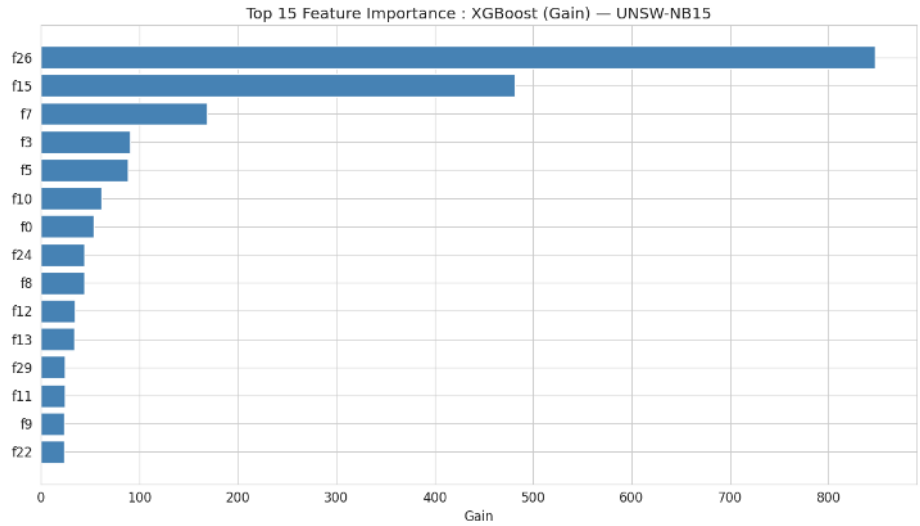


Figure 49: Feature importance by gain for XGBoost multiclass on UNSW-NB15.

Finally, the rapid convergence of the model with early stopping at iteration 61 out of 315 allowed rounds indicates a good fit to problem complexity without excessive overfitting. The validation weighted F1 of 0.8515 closely matches the test performance of 0.8496, confirming the validity of the model selection procedure.

4.5.3 Positioning of Baseline Models Against Deep Learning Architectures

Comparing the baseline models (RF, XGBoost) with the deep learning architectures (CNN, CNN-BiLSTM) yields important conclusions about the added value of deep learning for intrusion detection on UNSW-NB15.

In the binary setting, Random Forest and XGBoost achieve F1-Scores of 0.9885 and 0.9883 respectively, results that are directly comparable with, and in some metrics slightly superior to, the CNN (binary F1: 0.9617, Recall: 0.9407) and the CNN-BiLSTM (F1: 0.9728, Recall: 0.9594) on the same dataset. This observation is significant: on a moderately-sized dataset with careful preprocessing, well-optimised classical models can be competitive with deep learning, while offering better interpretability and lower inference computational cost.

However, a decisive advantage of DL architectures lies in their capacity to learn hierarchical representations directly from raw data, without requiring such intensive feature engineering. On larger datasets or those with stronger temporal dynamics, this advantage tends to increase.

In the 10-class multiclass setting, XGBoost achieves a macro F1 of 0.614, which is numerically higher than that of CNN (0.500) and CNN-BiLSTM (0.514), while also achieving a higher global accuracy of 84.16% against approximately 72% for both DL models. This difference is explained by the distinct weighting of classes in the two metrics: XGBoost performs better on intermediate classes such as Reconnaissance and Fuzzers, but faces greater difficulty on the smallest minority classes in absolute terms.

Overall, the two families of approaches present complementary profiles. Classical models offer better interpretability, faster convergence, and competitive performance on well-represented classes. Deep learning architectures are distinguished by their capacity to capture complex temporal and hierarchical patterns, at the cost of higher computational overhead and reduced decision transparency.

4.5.4 Statistical Significance Analysis

To determine whether the performance differences between the models were attributable to random cross-validation fold variation, the Wilcoxon signed-rank test (Wilcoxon Test, 2025) was applied to the macro F1-scores obtained from the five cross-validation folds for each configuration.

The test procedure is as follows. For each fold i, the difference between the CNN and CNN-BiLSTM models is computed as $di = F1_i^{CNN} - F1_i^{CNN-BiLSTM}$. These differences are then ranked in ascending order according to their absolute values, and a rank (R_i) is assigned to each difference. Subsequently, the ranks corresponding to the folds in which CNN outperforms CNN-BiLSTM (W^+) and those in which CNN-BiLSTM outperforms CNN (W^-) are summed separately. The test statistic is defined as $W = \min(W^+, W^-)$. A value of W approaching zero indicates that one model consistently outperforms the other across all folds.

To control the Type I error arising from conducting four simultaneous pairwise comparisons, the Bonferroni correction was applied, and the adjusted significance threshold was set to $\alpha_{corr} = \frac{0,05}{4} = 0,0125$. As shown in Table 22, none of the four configurations reached the adjusted statistical significance threshold (all comparisons yielded $p = 0.0625$ or $p = 0.1250$). This finding should be interpreted in light of the structural limitation imposed by the use of only $n = 5$ folds. Specifically, the minimum attainable p-value for the Wilcoxon signed-rank test with $n = 5$ is mathematically 0.0625, which inherently exceeds the Bonferroni-adjusted significance threshold. Nevertheless, the observed performance differences exhibited consistent directional trends. CNN achieved superior performance on the CIC-IDS2017 dataset for both tasks, whereas CNN-BiLSTM consistently produced higher macro F1-scores on the UNSW-NB15 dataset across both tasks. Statistical confirmation of these trends is therefore deferred to future studies employing larger evaluation budgets (e.g., $n \geq 10$ folds or repeated cross-validation).

Table 35: Wilcoxon Signed-Rank Test Results Comparing CNN and CNN-BiLSTM.

Configuration	CNN (mean ± SD)	CNN-BiLSTM (mean ± SD)	Δ	*p*-value	Significant?
CIC-IDS2017: Binary	0.9937 ± 0.0012	0.9918 ± 0.0005	+0.0019	0.0625	No
CIC-IDS2017: Multiclass	0.7008 ± 0.0317	0.6713 ± 0.0119	+0.0295	0.1250	No
UNSW-NB15: Binary	0.9617 ± 0.0009	0.9727 ± 0.0017	−0.0111	0.0625	No
UNSW-NB15: Multiclass	0.4999 ± 0.0064	0.5141 ± 0.0030	−0.0142	0.0625	No

4.6 Comparison with State-of-the-Art

Table 34 compares our results against recent NIDS studies on the same benchmarks. Direct numeric comparison across studies is not straightforward due to differences in preprocessing pipelines, train/test protocols, and reported metrics. In particular, FPR and throughput are not reported in most prior work. Within these constraints, our CNN model achieves competitive detection performance on CIC-IDS2017 for the binary task (Accuracy: 0.9975, F1-score: 0.9937, FPR: 0.0019, and throughput: 102,397 samples/s), approximately 4.7× higher than our CNN-BiLSTM model (21,828 samples/s) for binary classification.

On UNSW-NB15, the CNN-BiLSTM yields stronger binary classification performance (Accuracy: 0.9682, ROC-AUC: 0.9977) at the cost of significantly lower throughput, consistent with the accuracy–efficiency trade-off documented in recent comparative studies [dai2024network]. To the best of our knowledge, no prior work on these two datasets jointly reports FPR, ROC-AUC, detection time per sample, and throughput under a leak-free cross-validation protocol, making direct quantitative comparison with the full metric set impossible for most entries in Table 34.

Table 36: comparison with the literature.

Author(s) / Model	Dataset	Task	Acc.	Prec.	DR	F1	FPR	ROC-AUC	Det. Time	Thpt.	Split
CIC-IDS2017											
Song et al. [song2023csk] — CSK-CNN	CIC-IDS2017	B	99.94%	99.76%	99.96%	99.86%	—	—	42.59 s a	—	70/10/20
Song et al. [song2023csk] — CSK-CNN	CIC-IDS2017	M	99.80%	99.86%	99.80%	99.82%	—	—	33.60 s a	—	70/10/20
Talukder et al. [talukder2024machine] — RF	CIC-IDS2017	B	99.94%	99.94%	99.94%	99.94%	—	—	—	—	80/20 10-fold CV
Talukder et al. [talukder2024machine] — XGBoost	CIC-IDS2017	B	99.65%	99.65%	99.65%	99.65%	—	—	—	—	80/20 10-fold CV
Talukder et al. [talukder2024machine] — RF	CIC-IDS2017	M	99.99%	99.99%	99.99%	99.99%	—	—	—	—	80/20 10-fold CV
Talukder et al. [talukder2024machine] — XGBoost	CIC-IDS2017	M	99.94%	99.94%	99.94%	99.94%	—	—	—	—	80/20 10-fold CV
UNSW-NB15											
Jouhari & Guizani [jouhari2024efficient] — CNN-BiLSTM	UNSW-NB15	B	97.90%	97.91%	97.90%	97.90%	—	—	54.6 s b	—	80/20

Author(s) / Model	Dataset	Task	Acc.	Prec.	DR	F1	FPR	ROC-AUC	Det. Time	Thpt.	Split
Jouhari & Guizani [jouhari2024efficient] — CNN-BiLSTM	UNSW-NB15	M	97.09%	97.09%	97.62%	97.27%	—	—	442.20 s b	—	80/20
Song et al. [song2023csk] — CSK-CNN	UNSW-NB15	B	98.75%	90.06%	99.99%	94.76%	—	—	30.39 s a	—	70/10/20
Song et al. [song2023csk] — CSK-CNN	UNSW-NB15	M	92.05%	95.34%	92.05%	93.04%	—	—	80.81 s a	—	70/10/20
Talukder et al. [talukder2024machine] — RF	UNSW-NB15	B	99.59%	99.59%	99.59%	99.59%	—	—	—	—	80/20 10-fold CV
Talukder et al. [talukder2024machine] — XGBoost	UNSW-NB15	B	98.81%	98.81%	98.81%	98.81%	—	—	—	—	80/20 10-fold CV
Talukder et al. [talukder2024machine] — RF	UNSW-NB15	M	99.95%	99.95%	99.95%	99.95%	—	—	—	—	80/20 10-fold CV
Talukder et al. [talukder2024machine] — XGBoost	UNSW-NB15	M	95.04%	95.29%	95.03%	95.03%	—	—	—	—	80/20 10-fold CV
Dai et al. [dai2024network] — RF	UNSW-NB15	—	84.59%	—	92.24%	—	3.01%	—	—	—	10-fold CV
Dai et al. [dai2024network] — CNN-LSTM	UNSW-NB15	M	82.20%	—	82.41%	—	2.22%	—	—	—	10-fold CV

Author(s) / Model	Dataset	Task	Acc.	Prec.	DR	F1	FPR	ROC-AUC	Det. Time	Thpt.	Split
Dai et al. [dai2024network] — CNN-BiLSTM	UNSW-NB15	M	82.09%	—	92.51%	—	6.09%	—	—	—	10-fold CV
Dai et al. [dai2024network] — CNN-BiLSTM-Attn	UNSW-NB15	B	88.83%	—	98.51%	—	1.88%	—	—	—	10-fold CV
Our work — CIC-IDS2017, 5-fold cross-validation, 70/10/20 split for all tasks											
CNN (Ours)	**CIC-IDS2017**	**B**	**0.9975**	**0.9888**	**0.9987**	**0.9937**	**0.0019**	**0.9999**	**0.0098 ms/s**	**102,397 s/s**	**70/10/20 5-fold CV**
CNN-BiLSTM (Ours)	**CIC-IDS2017**	**B**	**0.9959**	**0.9857**	**0.9981**	**0.9918**	**0.0036**	**0.9999**	**0.0458 ms/s**	**21,828 s/s**	**70/10/20 5-fold CV**
CNN (Ours)	**CIC-IDS2017**	**M**	**0.9893**	**0.6661**	**0.9153**	**0.7008**	**0.0007**	**0.9951**	**0.0096 ms/s**	**104,091 s/s**	**70/10/20 5-fold CV**
CNN-BiLSTM (Ours)	**CIC-IDS2017**	**M**	**0.9630**	**0.6293**	**0.9083**	**0.6713**	**0.0011**	**0.9954**	**0.0326 ms/s**	**30,708 s/s**	**70/10/20 5-fold CV**
Our work — UNSW-NB15, 5-fold CV (DL) / 10-fold CV (XGBoost multiclass) / 70/10/20 split (ML binary)											
CNN (Ours)	**UNSW-NB15**	**B**	**0.9523**	**0.9638**	**0.9407**	**0.9617**	**0.0317**	**0.9942**	**0.0127 ms/s**	**80,055 s/s**	**70/10/20 5-fold CV**
CNN-BiLSTM (Ours)	**UNSW-NB15**	**B**	**0.9682**	**0.9865**	**0.9594**	**0.9728**	**0.0325**	**0.9977**	**0.0201 ms/s**	**49,853 s/s**	**70/10/20 5-fold CV**
CNN (Ours)	**UNSW-NB15**	**M**	**0.7208**	**0.4986**	**0.6671**	**0.4999**	**0.0283**	**0.9645**	**0.0117 ms/s**	**85,428 s/s**	**70/10/20 5-fold CV**

Author(s) / Model	Dataset	Task	Acc.	Prec.	DR	F1	FPR	ROC-AUC	Det. Time	Thpt.	Split
CNN-BiLSTM (Ours)	**UNSW-NB15**	**M**	**0.7649**	**0.5112**	**0.6321**	**0.5141**	**0.0276**	**0.9611**	**0.0454 ms/s**	**22,069 s/s**	**70/10/20 5-fold CV**
Random Forest (Ours)	**UNSW-NB15**	**B**	**0.9854**	**0.9950**	**0.9821**	**0.9885**	**0.0087**	**0.9989**	**0 ,0059 ms/s**	**170004,8461s/s**	**70/10/20**
XGBoost (Ours)	**UNSW-NB15**	**B**	**0.9852**	**0.9945**	**0.9822**	**0.9883**	**0.0096**	**0.9991**	**0,0003 ms/s**	**3796487,8874s/s**	**70/10/20**
XGBoost (Ours)	**UNSW-NB15**	**M**	**84,07%**	**86,88%**	**84,07%**	**61,17%**	**17 ,8%**	**97,05%**	**0,0003 ms/S**	**3073461,26s/s**	**10-fold CV**

B = Binary, M = Multiclass. '-' = not reported by the original authors, s/s = samples/s and ms/s = ms/sample/

[a] (Song et al., 2023)report total time (train + inference) over the full dataset, not per-sample detection time. Throughput in samples/s is not specified.

[b](Jouhari et al., 2024) report total train + prediction time over the full dataset. Per-sample detection time and throughput are not specified.

[c] (Dai et al., 2024) report only Accuracy, DR, and FPR. Dataset split not specified in the original paper.

[d] (Talukder et al., 2024) report identical Accuracy, Precision, Recall and F1-scores, suggesting these are weighted averages on balanced evaluation sets. ROC-AUC, FPR, detection time and throughput are not reported.

[†] All our DL models (CNN, CNN-BiLSTM) use 5-fold CV with quantile-based transformer and SMOTE applied strictly within each training fold to prevent data leakage. Our ML baselines (RF, XGBoost) use a fixed 70/20/10 stratified split with intra-fold target encoder on UNSW-NB15.

[‡] Differences in preprocessing pipelines, feature engineering, train/validation/test protocols, and dataset versions limit strict cross-study comparability.

[§] Throughput was not measured for the Random Forest and XGBoost models, as the experimental setup prioritized classification performance metrics; inference time profiling was not instrumented during the evaluation phase.

Chapter 5

5 Conclusion and Future Work

This thesis has investigated the application of deep learning architectures to the problem of network intrusion detection, evaluating two model families, CNN and CNN-BiLSTM, across two benchmark datasets, CIC-IDS2017 and UNSW-NB15, under both binary and multiclass classification settings. The experimental evaluation was grounded in a rigorous and reproducible methodological framework: a unified preprocessing pipeline including Focal Loss training, and a comprehensive set of evaluation metrics. Two classical machine learning baselines, Random Forest and XGBoost, were additionally evaluated on UNSW-NB15 to provide a comparative reference for the deep learning results.

5.1 Answers to Research Questions

Regarding RQ1, both CNN1D and CNN-BiLSTM achieve high accuracy and strong discriminative performance in binary classification, with ROC-AUC values consistently above 0.994 across all configurations. In multiclass classification, performance degrades as expected, with macro F1-Scores between 0.500 and 0.714, reflecting the inherent difficulty of fine-grained attack-type discrimination under severe class imbalance.

Regarding RQ2, the CNN-BiLSTM provides a meaningful advantage over CNN1D specifically in the binary UNSW-NB15 setting, where temporal pattern complexity is higher. On CIC-IDS2017, the advantage is marginal in binary mode and reverses in multiclass mode, where the CNN1D is more stable and achieves a higher macro F1. The CNN-BiLSTM advantage is therefore task-dependent and dataset-dependent, and it comes at a consistent computational cost of 4 to 6 times higher inference latency.

Regarding RQ3, classical ML baselines are competitive with deep learning in binary detection on UNSW-NB15, with XGBoost in particular achieving equivalent F1 performance in a fraction of the training time. Deep learning models do not demonstrate a clear advantage at this scale and with this preprocessing pipeline, suggesting that the benefit of DL becomes more pronounced at larger scale or with raw input representations that do not require feature engineering.

5.2 Limitations

Several limitations of this study deserve explicit acknowledgement. First, the evaluation is conducted exclusively on two static benchmark datasets. While CIC-IDS2017 and UNSW-NB15 are among the most widely used reference benchmarks in the NIDS literature, they represent network traffic captured in controlled laboratory environments and may not fully reflect the diversity and volatility of real-world production traffic. Second, the models were not evaluated on streaming data, meaning that the temporal dynamics of live network monitoring, including concept drift and distribution shift, are not addressed in this work. Third, the hyperparameter configurations, while carefully tuned, were not jointly optimised across all model-dataset combinations through a unified automated search, which may leave marginal performance gains unrealised. Finally, the feature engineering pipeline, while leak-free and methodologically sound, involves a degree of domain-informed preprocessing that may not transfer directly to raw packet-level or flow-level data without adaptation.

5.3 Future Work

Several directions for future research follow naturally from the findings and limitations of this thesis.

The most immediately impactful avenue concerns minority class detection. The consistent failure of all evaluated models on rare attack categories motivates the development of more targeted strategies, including class-conditional SMOTE with varying oversampling ratios, cost-sensitive loss functions with per-class weight tuning, ensemble methods that combine specialized binary classifiers per attack family, and few-shot learning approaches that can generalize from very limited examples.

A second direction concerns threshold optimization. The results show that all evaluated models, including those with poor per-class F1-Scores, maintain high ROC-AUC values at the macro level. This indicates that the probability outputs are informative, but that the default 0.5 threshold is suboptimal for imbalanced multiclass settings. Class-specific threshold calibration using validation data, guided by operational cost functions that reflect the relative severity of different attack types, could yield substantial improvements without modifying the model architecture.

A third direction involves the evaluation of more advanced architectures. Transformer-based models, which have shown strong performance on sequential data in natural language processing and time-series forecasting, represent a natural extension of the BiLSTM component evaluated here. Self-attention mechanisms could potentially capture long-range

dependencies in network flow sequences more effectively than recurrent units, particularly in datasets with richer temporal structure.

6 References

Alashjaee, A. M., & Alqahtani, F. (2025). Enhanced intrusion detection system IoT network security model by feed forward neural network and machine learning. *Scientific Reports*, *15*(1), 36085.

Brown, L. V. (2008). *Computer security : Principles and practice*. Pearson Prentice Hall.

Coco, A., Dias, T., & van Benthem, T. (2022). Illegal : The SolarWinds Hack under International Law. *European Journal of International Law*, *33*(4), 1275-1286. https://doi.org/10.1093/ejil/chac063

Colonial Pipeline : The DarkSide Strikes. (s. d.). [Legislation]. Consulté 2 mai 2026, à l'adresse https://www.congress.gov/crs-product/IN11667

Dai, W., Li, X., Ji, W., & He, S. (2024). Network intrusion detection method based on CNN-BiLSTM-attention model. *IEEE access*, *12*, 53099-53111.

Farhan, M., Waheed Ud Din, H., Ullah, S., Hussain, M. S., Khan, M. A., Mazhar, T., Khattak, U. F., & Jaghdam, I. H. (2025a). Network-based intrusion detection using deep learning technique. *Scientific Reports*, *15*(1), 25550.

Farhan, M., Waheed Ud Din, H., Ullah, S., Hussain, M. S., Khan, M. A., Mazhar, T., Khattak, U. F., & Jaghdam, I. H. (2025b). Network-based intrusion detection using deep learning technique. *Scientific Reports*, *15*(1), 25550.

Henry, A., Gautam, S., Khanna, S., Rabie, K., Shongwe, T., Bhattacharya, P., Sharma, B., & Chowdhury, S. (2023). Composition of hybrid deep learning model and feature optimization for intrusion detection system. *Sensors*, *23*(2), 890.

Jouhari, M., Benaddi, H., & Ibrahimi, K. (2024). Efficient intrusion detection : Combining x 2 feature selection with CNN-BiLSTM on the UNSW-NB15 dataset. *2024 11th International Conference on Wireless Networks and Mobile Communications (WINCOM)*, 1-6. https://ieeexplore.ieee.org/abstract/document/10658099/

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*. https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. *Proceedings of the IEEE international conference on computer vision*, 2980-2988. http://openaccess.thecvf.com/content_iccv_2017/html/Lin_Focal_Loss_for_ICCV_2017_paper.html

Magazine, C. (2018, février 21). Cybercrime To Cost The World $10.5 Trillion Annually By 2025. *Cybercrime Magazine*. https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016/

Moustafa, N., & Slay, J. (2016). The evaluation of Network Anomaly Detection Systems : Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective*, *25*(1-3), 18-31. https://doi.org/10.1080/19393555.2015.1125974

Peng, H., Wu, C., & Xiao, Y. (2023). CBF-IDS : Addressing class imbalance using CNN-BiLSTM with focal loss in network intrusion detection system. *Applied Sciences*, *13*(21), 11629.

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, *1*(2018), 108-116.

Song, J., Wang, X., He, M., & Jin, L. (2023). CSK-CNN : Network intrusion detection model based on two-layer convolution neural network for handling imbalanced dataset. *Information*, *14*(2), 130.

Talukder, Md. A., Islam, Md. M., Uddin, M. A., Hasan, K. F., Sharmin, S., Alyami, S. A., & Moni, M. A. (2024). Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction. *Journal of Big Data*, *11*(1), 33. https://doi.org/10.1186/s40537-024-00886-w

Türk, F. (2023). Analysis of Intrusion Detection Systems in UNSW-NB15 and NSL-KDD Datasets with Machine Learning Algorithms. *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*, *12*(2), 465-477. https://doi.org/10.17798/bitlisfen.1240469

Udurume, M., Shakhov, V., & Koo, I. (2024). Comparative analysis of deep convolutional neural network—Bidirectional long short-term memory and machine learning methods in intrusion detection systems. *Applied Sciences*, *14*(16), 6967.

Vinayakumar, R., Soman, K. P., Prabaharan Poornachandran, & Akarsh, S. (2019). Application of Deep Learning Architectures for Cyber Security. In A. E. Hassanien & M. Elhoseny (Éds.), *Cybersecurity and Secure Information Systems* (p. 125-160). Springer International Publishing. https://doi.org/10.1007/978-3-030-16837-7_7