Faculty of Media Engineering and Technology
German University in Cairo

A Deep Learning Approach for Intelligent Intrusion Detection

Bachelor Thesis

Author:	Ziad Amgad
Supervisors:	Dr. Gamal A. Ebrahim
Submission Date:	29 May, 2025

Faculty of Media Engineering and Technology
German University in Cairo

A Deep Learning Approach for Intelligent Intrusion Detection

Bachelor Thesis

Author:	Ziad Amgad
Supervisors:	Dr. Gamal A. Ebrahim
Submission Date:	29 May, 2025

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowledgment has been made in the text to all other material used

Put your name here
29 May, 2025

Abstract

The growing complexity of cyber threats calls for network security adaptation that goes beyond conventional Intrusion Detection Systems (IDS), which try in vain to comply with new, sophisticated attacks and comprehensive data. Deep learning provides a great perspective for improving IDS through automatic learning of complex feature representation from network traffic. This thesis is inspired by the necessity to systematically analyze different deep learning architectures for intelligent intrusion detection to get information on their practical usage and performance.

Previous studies about deep learning in the context of IDS have experimented with different models, but extensive comparative studies under uniform systematic preprocessing—particularly addressing extreme class imbalance and stringent feature selection—are less encountered. There are still voids in explaining the exact strengths and weaknesses of different architectures when differentiating an enormous set of fine-grained attack categories. Furthermore, the overall trade-offs of model complexity, detection accuracy, and computational efficiency are not always explicitly defined. This study will seek to achieve these through a rigorous, comparative analysis.

This research tests and compares four deep learning models: a 2D Convolutional Neural Network (2D CNN), a Multi-Layer Perceptron (MLP), TabNet, and a hybrid CNN-Bidirectional Long Short-Term Memory (CNN-BiLSTM) network, on the preprocessed UNSW-NB15 benchmark dataset for both multi-class and binary (Normal vs. Attack) classification. The findings indicate a clear performance hierarchy in multi-class detection, where the CNN-BiLSTM achieved the highest accuracy of 91.3% and a macro F1-score of 0.900, followed by TabNet with 85.7% accuracy and a 0.831 macro F1-score. The 2D CNN and MLP models yielded accuracies around 81.5-81.6%. In the binary task, all models performed exceptionally well, with the CNN-BiLSTM reaching 98.9% accuracy. However, most models, apart from the CNN-BiLSTM, faced difficulties reliably distinguishing similar attack types in the multi-class context. A distinct trade-off between model complexity, classification performance, and computational efficiency was observed, with the hybrid CNN-BiLSTM offering superior accuracy at the cost of longer training time (approx. 100 minutes), while simpler models like TabNet trained faster (approx. 24 minutes) but with comparatively lower peak accuracy. This work highlights the critical role of architecture selection and identifies key areas for enhancing fine-grained attack discrimination.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With more devices being networked and digital infrastructure becoming indispensable, Cybersecurity has become more of a major global worry for institutions and people. Greater connectivity, therefore, implies greater vulnerability, thus resulting in the growth of frequency, sophistication, and severity of cyber attacks [1]. Intrusion Detection Systems (IDS) play a key role in their discovery and response to such activity, as important components of an effective security posture. Conventional IDS methodologies are, however, plagued with serious shortcomings in the ability to detect novel threats and cope with sheer volume and diversity of existing network traffic. In response, researchers have turned in greater numbers to sophisticated machine learning, and particularly deep learning, techniques to more effectively empower IDS. This thesis examines the use of various deep learning architectures for the network intrusion detection task with the aim of evaluating their performance and comparing their outcomes in this demanding scenario.

1.1 Problem Statement

The underlying problem is the proper and effective detection of different network intrusions by utilizing advanced computational techniques, that is, deep learning. Traditional signature-based IDS are inherently limiting since they are dependent on detecting known attack signatures, hence are helpless against yet unknown or dynamic attacks. While anomaly-based detection systems can effectively identify deviance from clearly defined normal behavior, they are often weak in the presence of high false positives, drowning security analysts and potentially obscuring genuine incidents. Moreover, successful modern IDS design is also beset by several inherent challenges. Network traffic data is typically high-dimensional and complex with numerous features having potentially complex, nonlinear interactions. Datasets are typically highly imbalanced classes with regular traffic heavily dominating malicious traffic, and some attacks being very much less frequent than others, making it hard to train balanced models. Proper feature representation matters, but raw network data might be hard to process to make it learning-friendly. As the sophistication of the evasion methods of attackers increases, detection systems must be agile and powerful in reaction. Finally, even advanced models can struggle to differ-

entiate between certain types of attacks with slight differences or with inter-inherently overlapping feature representations. Deep learning methods are discussed in this thesis in the context of addressing these inherent drawbacks in network intrusion detection.

1.2 Motivation

The impetus for this project arises from the urgent need for more effective network security measures that can counter the increasing and changing nature of cyber attacks. The known shortcomings of current IDS methods call for the investigation of new techniques, such as deep learning, that have shown promise in learning sophisticated patterns and evolving to new attack vectors. Deep learning is particularly promising since it has an inherent ability to learn hierarchical feature representations automatically from large amounts of data. Such capability can potentially outperform the limitations of handcrafted feature engineering and allow for the detection of subtle indicators of malignant behavior that would be overlooked by other methods.

However, applying deep learning to successfully identify intrusions is a challenge that involves overcoming great complexity. There are many different deep learning architectures, each with varying theoretical advantages and disadvantages compared to the kind of patterns it is well-suited to detect. Therefore, there is an urgent need to critically compare and contrast the actual performance of different types of deep learning models applied to standard intrusion detection tasks on standardized data sets. Additionally, deep insight into what data preprocessing procedures do, including those that are used to manage class imbalance is essential. It is also necessary to understand which attack types are typically well-handled by existing deep learning methods and which continue to be challenging, thereby guiding future research. Assessing the intrinsic trade-offs in model performance, complexity, and computational resource needs is also essential as these impact the deployability of real-world applications substantially. Driving this work is the need to cover these gaps with the aim of delivering pragmatic advice on successful deep learning adoption for intrusion detection capability improvement.

1.3 Project Objectives

The overall objective of this thesis is to compare the efficacy of various deep learning approaches when applied to the multi-class network intrusion detection problem. This entails an examination of the process from data preparation through model training and evaluation, seeking to identify comparative strengths and weaknesses of competing architectural paradigms in this application of cybersecurity. The study aims to illuminate the real-world uses and intricacies of using these new models in network threat detection.

To achieve this overarching goal, several specific goals guide this work. The research aims to comprehensively investigate means by which deep learning techniques can be utilized to classify types of network attacks as well as normal network traffic. A central component of this involves performing extensive preprocessing on an appropriate

benchmark dataset, methodically addressing common data quality issues of imbalance, redundancy, and outliers, to prepare the data for effective model training. Subsequently, the project involves the installation, training, and rigorous optimization of a variety of different deep learning architectures chosen for their differing approach to feature learning. All of the models trained are then systematically evaluated by using appropriate classification metrics to provide an exhaustive evaluation of their accuracy and reliability for different classes. Based on these per-model evaluations, a comparison is drawn to contrast the models based on overall accuracy, robustness to class imbalance, training computational efficiency, and special prowess in detecting particular types of intrusions. Finally, the research attempts to establish and discuss general challenges for the models evaluated, particularly with regards to the fine-grained discernment between challenging attack types, thus highlighting areas for future improvement.

1.4 Thesis Organization

The organization of this thesis is the following. Chapter 1 has an introduction to the research area, a problem definition under consideration, the motivation for the work, and its objectives. Chapter 2 provides the background information required, with information regarding intrusion attacks, detection mechanisms, and relevant machine learning and deep learning theory, along with a summary of the relevant work in the area. Chapter 3 describes the methodology employed in this research, including the choice of the dataset, the preprocessing steps, and the individual deep learning models employed and compared. Chapter 4 presents the experimental results and a comparative analysis of the model results. Chapter 5 is the final chapter of the thesis, summarizing the key findings and suggestions for future work.

The main components of this thesis are:

- **Background:** Provides an overview to intrusion detection, describes fitting technologies including deep learning approaches, and discusses relevant literature of the project currently available.
- **Methodology:** Explains in minute detail how the project was carried out, from pre-processing of data, through the actual deep learning models used, their details, and also how testing was done, to make the work understandable and even replicable.
- **Results:** Explains the findings of the experiments, setting out the performance metrics for all models and giving a comparative analysis of their efficiency and character.
- **Conclusion and Future Work:** Offers a brief discussion outlining the success of this paper, the key points gained from the findings, and potential avenues of future work in an effort to move forward from these findings and continue to hone intrusion detection capacity.

Chapter 2

Background

The rise in importance of cybersecurity stems from the fact that cyber threats are on the rise both in scope and magnitude. The concern grows, especially because of the multitude of threats such as denial-of-service (DoS) attacks, malware exploits, malicious insiders, and zero-day vulnerabilities or insider threat that affect not only organizations but also governments and individual users. To combat all these, intrusion activity in a host or network environment can be identified, processed, and countered by employing an Intrusion Detection System (IDS). Application of different Deep Learning techniques for IDS frameworks based on Convolutional Neural Networks (CNN), Long Short Term Memory Networks (LSTM), Graph Neural Networks (GNN), etc., has facilitated increased flexibility, speed and accuracy in threat detection. Here, deep intrusions and deep learning are discussed along with other features of the landscape of intrusion attacks and IDS approaches that seek to achieve the goals, which are the foundation of the research and methods provided in this research.

2.1 Attacks

Compromising a system's integrity, confidentiality, and availability is effortlessly done when modern computer networks are intruded. Security attacks start with straightforward unauthorized incidents before advancing to sophisticated cyber break-ins which attempt security feature bypassing. Sophisticated and evolved methodologies and characteristics comprise diverse types of intrusion attacks.

2.1.1 Denial of Service (DoS) and Distributed Denial of Service (DDoS) Attacks

The objective of DoS and DDoS attacks is to disrupt and incapacitate network resources such that the target opponent is unable to make use of it. Attackers send far too many requests for a service to a host such that its bandwidth as well as processing power is completely consumed. The more sophisticated form, DDoS, uses botnets to allocate the

attack in different regions, which makes it difficult to reduce the strength of the assault [2].

2.1.2 Analysis

Analysis traffic comes from tools that analyze and capture network packets. Although their use can be justified for resolving problems concerning the network, from a security perspective, such tools are often used to monitor network traffic, capture sensitive data such as credentials, or appreciate a network's architecture, its defensive mechanisms, and its oversight strategies for devising subsequent attacks [3].

2.1.3 Backdoor

A backdoor provides a bypass for a computer system or network login alongside avoidance of the normal username, password, and multifactor authentication requirements. After compromising a vulnerability, Attackers frequently embed concealed backdoors within software or Trojans in order to sustain, unnoticed, access to the system. Information systems are remotely controlled by sophisticated attackers for the purpose of stealing confidential information, infecting the computer with malicious programs, and executing subsequent cyber assaults.

2.1.4 Worms

Worms are a type of malware which can self replicate and spread across computer systems by exploiting vulnerabilities within the Operating system and Applications. Unlike viruses, they do not require any human action or the implantation into existing software for themselves to be released. The main objective is to disseminate as fast as they can while consuming valuable resources of the system and network in the process. They are known to carry malignant cargo which is designed to erase data, create backdoors to intrusion, or conscript the infected device into a botnet.

2.1.5 Exploits

Exploits tend to use sophisticated coded algorithms and sequences which are created to take advantage of software, or network system, or hardware vulnerabilities. Most cyber-criminals use these exploits for causing unintended behaviors like access-hacking, malware execution, privilege escalation, and denial of service attack which breach the specified security systems.

2.1.6 Fuzzers

Fuzzers are tools in which the boundaries of certain applications are probed by feeding large amounts of nonsensical, random or unusual data. The objective is to examine and determine whether an application suffers an accident, hang or maloperation in specific

conditions. Often, these odd behaviors are a symptom of far more complicated issues, such as (but not limited to) buffer overflows that malicious attackers could evaluate and construct exploit techniques to gain systemic control.

2.1.7 Generic

The Generic classification is frequently a parent label for malicious payloads associated with Trojans that do not fall within clearly delineated categories of attacks such as Worms or Shellcode. Data moving through these networks tend to have their origins from computers or devices that are possessed allowing personal information to be stolen or planning coordinated attacks or joining massive botnet systems.

2.1.8 Shellcode

A typical shellcode is a small and executable code used as a payload post system exploit. It gets its name from its primary purpose placing a command shell on the targeted machine. Without this shell friendly individuals can take over the system, execute commands, alter files, and execute dangerous procedures.

2.1.9 Reconnaissance

Reconnaissance attacks refer to attacks in which, attackers will try to get information pertaining to their targets before selecting to carry a full attack [4]. Techniques include network scanning to expose active hosts, open port and active running service scanning (port scanning), and network structure mapping. Target environment is identified, vulnerabilities of interest exposed, and best-suited attack methods chosen to exploit to follow.

2.2 Intrusion Detection System

Because they can pinpoint hostile activities and policy violations, intrusion detection systems (IDS) are a pivotal part of cybersecurity due to hostile activity surveillance as well as system activity monitoring. As cybersecurity complications increase, IDS systems are continuously evolving from basic, rule-based technologies, to machine learning and deep learning techniques [5]. Prior to the happening of adverse effects, their core goal is to find security flaws such as, but not limited to, malware infections, denial of service (DoS) assaults, and illegal access.

The two basic types of intrusion detection systems are network-based (NIDS) and host-based (HIDS) systems. NIDS keep an eye on network traffic for questionable activity and HIDS examine system logs, file integrity and application behavior [6]. Both detection methodologies, conventional IDS techniques, are based on signature and anomaly. As long as there is a well-documented threat, detection will more than likely succeed, however, signature-based detection tends to struggle with so called, zero-day attacks. In regard to

detection, anomaly-based relies on the use of machine learning and statistical models to enhance threat detection that hasn't been recognized yet [7].

The effectiveness and precision of IDS systems have increased due to latest developments in machine learning. Convolutional and recurrent neural networks (RNNs) are two examples of deep learning models that improve feature extraction and classification tasks [8]. Even deeper fusion models that add random forests or ensemble learning to deep learning techniques improve detection performance further [9]. Some techniques like CGFL federated learning are able to strengthen the IDS while keeping the data privacy safe in distributed cases [6].

As illustrated in Figure 2.1, these advancements collectively form a layered and intelligent IDS capable of responding to dynamic and unknown threats. These advances have not removed the obstacles of high false positive rates, data imbalance, and real-time processing in large-scale networks for most IDS systems [10]. New designs look at ways such as using generative adversarial networks (GANs), attention mechanisms, and graph neural networks (GNNs) to enhance IDS capabilities [11]. The scientific research on cybersecurity is still vigorously focusing on building hyper-intelligent and hyper-adaptive IDS because the cyber attack vectors are endless.

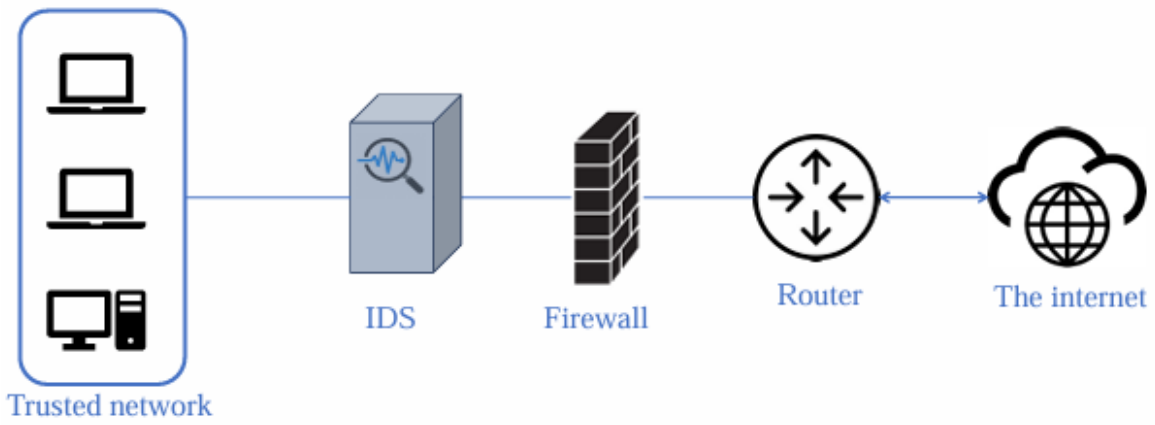


Figure 2.1: The IDS model diagram [11]

2.3 Deep Learning in Intrusion Detection

Incorporating sophisticated neural network structures into deep learning systems of an IDS has made its accuracy and efficiency better. This chapter focuses on the six most popular models of deep learning which are used in IDS: Convolutional Neural Network (CNN), Long Short-Term Memory Networks (LSTMs), Deep Neural Networks (DNNs), Multi Layer Perceptrons (MLPs), Gated Recurrent Units (GRUs), and Graph Neural Networks (GNNs). In addition, each type has distinct network traffic data patterns called spatial, temporal, or relational dependencies that help with threat classification.

2.3.1 Convolutional Neural Networks (CNNs)

Their proficiency in automatically obtaining multi-level features from network traffic data explains the popularity of CNNs in Intrusion Detection Systems. CNNs use multiple convolutional layers to take advantage of the spatial and temporal correlations in the data, which is important in intrusion detection [12]. CNN utilized models for intrusion detection systems use kernel filters along with pooling parts for dimensionality reduction while maintaining the relevant details concerning the encoded threats. Such models are useful in the examination of packet-based network traffic patterns for detecting anomalies [9].

2.3.2 Long Short-Term Memory Networks (LSTMs)

An improved version of an RNN in the guise of an LSTM was created to cope with sequential data with long dependencies. LSTMs are suited for IDS uses since they exhibit good performance in the capability of shifting network traffic patterns through time. Memory cells and gate structures allow LSTMs to curb the vanishing gradient problem, allowing the model to learn through sequences of patterns of attacks which are extremely lengthy [8]. LSTM-based IDS models have been shown to be the most performing ones for detecting advanced cyber threats with sequential dependencies [13].

2.3.3 Deep Neural Networks (DNNs)

In order to evaluate raw network traffic data and discover intricate representations to detect intrusions, DNNs are made up of several fully connected layers. The models are well-known for their ability to extract high-dimensional features and detect attacks with high accuracy [5]. DNN-based IDS systems are likely to include attention mechanisms and autoencoders in an attempt to improve feature learning and detection effectiveness [10]. The flexibility of DNNs allows them to be combined with other machine learning techniques, e.g., reinforcement learning, in order to tune the detection performance [8].

2.3.4 Multi-Layer Perceptrons (MLPs)

Multi-layer feed-forward neural networks in MLPs are those whose each neuron in a layer is connected to all neurons in layers immediately on its other side. MLPs are applied most in Intrusion Detection Systems (IDS) because they can represent intricate non-linear relationships in order to model traffic data. A single neuron in an MLP imposes an activation function in which one may preprocess the input such that a hierarchical attack representation could be learned by the network. Unlike CNNs and LSTMs, MLPs do not learn anything spatial or temporal; therefore, they work well when feature extraction is finished beforehand. MLP based IDS models generally include feature selection and some dimensionality reduction methods for the purpose of improving classification accuracy and reducing processing time [5].

2.3.5 Gated Recurrent Units (GRUs)

The special variant of RNNs, GRUs, is designed to manage sequential data properly and avoid the problem of vanishing gradients. They are very useful in intrusion detection systems for capturing the time dependency of network traffic, just like LSTMs, but with a lesser complex structure. GRUs have update and reset gates that control the information flow, enabling the model to sustain long-term dependencies with less parameters than LSTMs. This improves computational efficiency of the GRU-based IDS models without significantly weakening the detection of attacks which are sequential in nature over a period of time [8].

2.3.6 Graph Neural Networks (GNNs)

GNNs have received a lot of attention in the area of intrusion detection system research because of their ability to model complex dependencies in network traffic data. GNNs differ from conventional deep learning models in that they capture cybersecurity information in the form of graphs, which has both spatial and relational attributes within a network [11]. As shown in Figure 2.2, GNN-based IDS systems use graph embeddings and attention-based methods to identify abnormal activities in huge networks. Recently proposed feature aggregation and graph convolution methods in GNNs have been shown to greatly improve the accuracy of intrusion detections [6].

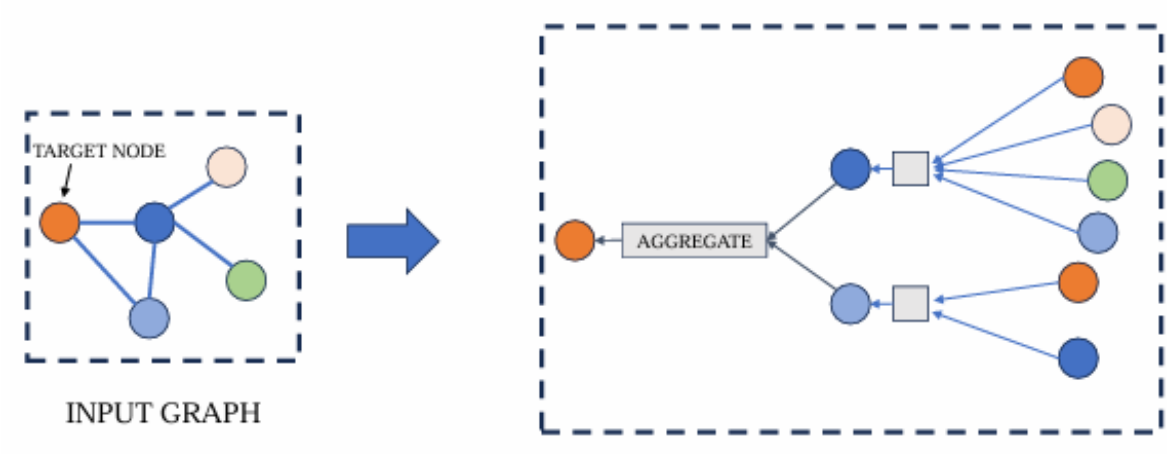


Figure 2.2: GNN's message-passing system [11]

2.4 Machine Learning in Intrusion Detection

There is a range of machine learning techniques that have improved the ability of IDS, particularly by improving the detection and categorization of online security events. In this subsection, a description is made of important machine learning techniques such as

Random Forest (RF), Support Vector Machines (SVM), k-Nearest Neighbors (KNN), and others.

2.4.1 Random Forest (RF)

An important technique of machine learning is Random Forest which is one of the ensemble methods of learning. Building multiple decision trees and combined their outputs to obtain a better classification is what Random Forest is. The essence of RF is the training of every tree on random features and samples striving to overcome overfitting and enhance generalization [9]. In IDS, RF is frequently used to classify network intrusions due to its strength in coping with high dimensionality and identifying sophisticated attacks. It has been implemented to classify network intrusions using feature importance and voting [5].

2.4.2 Support Vector Machines (SVM)

SVM, or Support Vector Machine, is a complex supervised learning technique which splits classes of data located in multidimensional space by constructing an optimal hyperplane. SVM is considered the best option for intrusion detection systems because it accommodates data that cannot be linearly separated through the use of kernel functions [6]. SVM based IDS models have been used to detect abnormal behavior in computer networks by elevating the input features to higher dimensional manifolds which improves accuracy in the presence of sophisticated cyber attacks [7].

2.4.3 k-Nearest Neighbors (KNN)

KNN, or K Nearest Neighbor, is one of the easiest supervised learning algorithms which predicts a target class for a new data point based on the class of its k nearest neighbors. It is especially helpful in intrusion detection systems for immediate anomaly detection because there are no prerequisites in terms of training the KNN classifier, and it is able to assimilate to new forms of attacks at once [10]. KNN has been used in intrusion detection system architectures in order to classify traffic anomalies based on distance or other similarity measures to achieve higher flexibility in the detection of intrusions [8].

2.5 Reinforcement Learning in Intrusion Detection

RL represents a solution for IDS due to its ability to automatically learn cyber threat patterns without requiring human supervision. RL functions through the interaction of an agent and environment where the agent improves his actions and learns through experience [14].

Deep Q-Learning (DQL) is also widely used in RL for IDS as it estimates Q-values using neural networks and uses them to select actions in threat detection [14]. The combination of deep reinforcement learning and GANs in models like DRL-GAN is a

solution to address data bias and generalize problems in the context of intrusion detection [15].

Furthermore, advanced methods employing deep autoencoders with optimal policy selection have performed well in the classification of attacks on IDS [8]. These algorithms leverage reinforcement learning (RL) to learn dynamically adapt to evolving threats, hence improving real-time detection accuracy. Merging deep autoencoders into a reinforcement learning framework, as depicted in Figure 2.3, is an optimistic architecture for intrusion detection within advanced and dynamic networks.

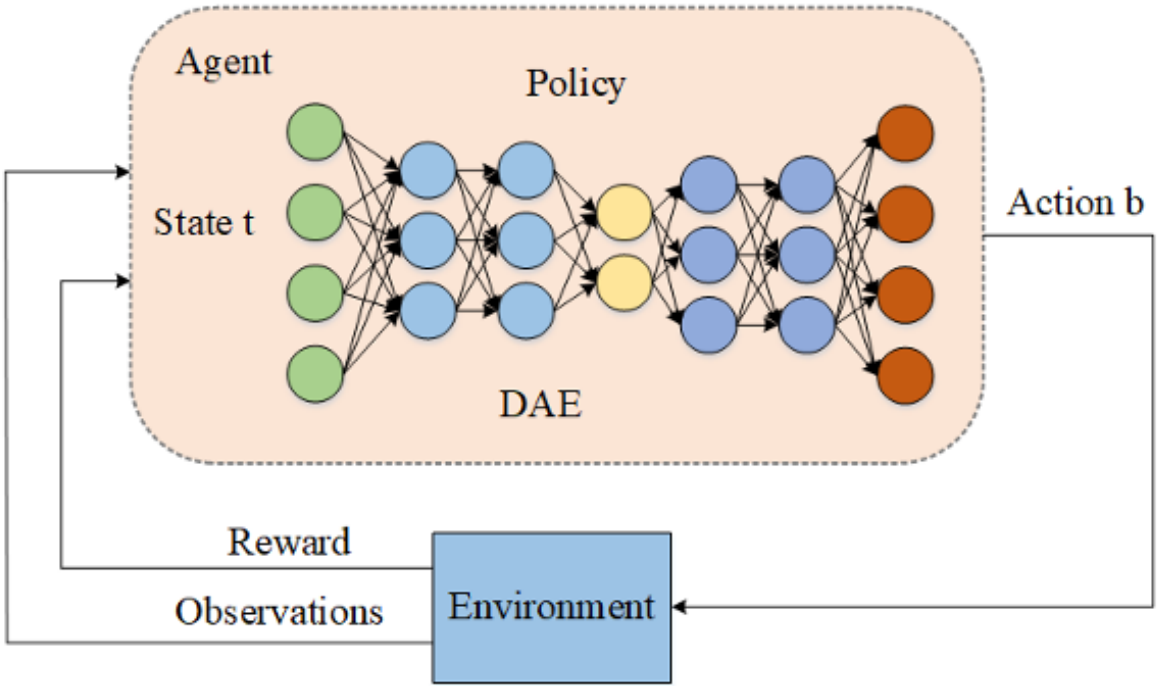


Figure 2.3: Improved deep reinforcement learning architecture [8]

2.6 Literature Survey

This part examines the different existing studies on the topic of IDS, underscoring three categories: techniques based on machine learning, approaches based on reinforcement learning, and those applying deep learning. Each subsection elaborates pertinent literature by explaining its methods, contributions, and relevance to the overall performance of IDS.

2.6.1 Reinforcement Learning Approaches

Alavizadeh and colleagues created an intrusion detection system utilizing a novel Deep Q-Learning (DQL) model through the application of reinforcement deep learning methods [14]. The primary concept is that such systems can autonomously develop optimal

responses to cyber-attacks because a deep learning model capable of estimating Q-values is implemented. In order to enhance a user's intelligent behavior, the model captures the network traffic, thus allowing for the improvement of automatic intrusion detection in real-time. The study shows how DQL reduce false alarms and improve threat detection in an flexible synthetic intelligence-based security system.

Strickland in [15] developed a new model for network intrusion detection, DRL-GAN, which integrates Deep Reinforcement Learning and Generative Adversarial Networks (GANs). The system is capable of accommodating new attack patterns because of its DRL component, which constantly refines its defensive strategies. At the same time, class imbalance is addressed by the GAN which creates synthetic attack samples to aid training. Their research shown that using DRL-GAN enhances detection efficacy in both binary and multiclass classification tasks, the results show fewer false positives and improves the overall integrity of the IDS.

Roy and Kalita [8] developed a new deep autoencoder-based reinforcement learning model with improved flamingo search policy selection for attack classification. The use of deep autoencoders and network traffic data makes it possible to discover vital attributes, diminish noise, and improve the entire process of feature selection. This is followed by the application of reinforcement learning for dynamic policy selection, making the model more adaptable to the changing conditions in which intrusion detection systems must operate. High accuracy of the classification is obtained here at the expense of small computational resources.Notably the improved resource efficiency provided by deep autoencoders is helping allow for the greater implementation of IDS systems upon expansive cybersecurity infrastructures. The overall flow of this attack classification and response approach is illustrated in Figure 2.4.

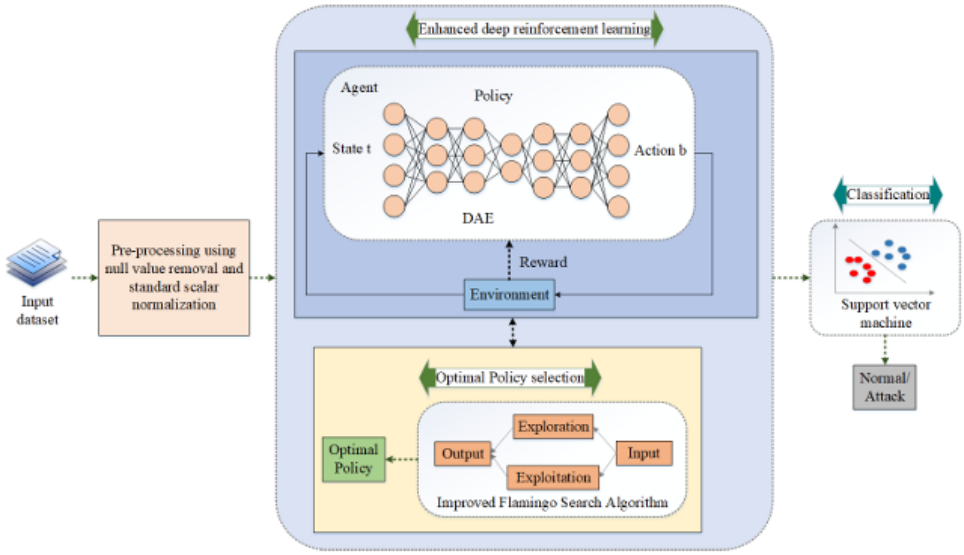


Figure 2.4: Flow diagram of the proposed attack classification and response scheme [8]

2.6.2 Machine Learning Approaches

Mari [16] used GANs to test how well an ML-based intrusion detection system (IDS) performed. In their case, the GANs-based model created synthetic attack samples for the classifying models to work with during training. An intrusion detection system (IDS) combined with imposer learning techniques is used mitigate data imbalance and strengthen the defenses against cyberattacks.

Musleh [17] examined quanta extraction methods for intrusion detection systems in Internet of Things (IoT) settings. The detection of IoT network anomalies happened through the execution of machine learning algorithms including logistic regression and SVMs. Their work showed that automation feature selection considerably increases an IDS's effectiveness and agility, thus making it more applicable in real-time security scenarios.

Ali [5] compared the use of deep learning models and traditional machine learning models for intrusion detection systems (IDS) problems. Their study included several machine learning procedures such as decision trees, support vector machines (SVM), and k-nearest neighbor (KNN) that were emphasized on their capabilities to detect intrusions in the network. It was observed that deep learning models had more accurately, but traditional machine learning models were still valuable for lower computational processes and their ease of implementation.

Nassreddine [7] used ensemble learning that combines different classifiers based on correlation and embedded feature selection methods for IDS. To improve the detection rate and decrease overfitting from their classification algorithm, their model involved random forest and multi-classing gradient boosting. It was proven in their study that ensemble techniques have the ability to improve the robustness of IDS without compromising the detection accuracy of evolving cyber attacks.

2.6.3 Simple Deep Learning Approaches

Ali [5] studied the deep learning approaches for IDS systems comparably with other machine learning techniques. They noted that the usage of modern neural networks deep learning model is much more efficient compared to traditional techniques, when it comes to high dimensional data and complicated attack patterns. This research proved that deep learning technology achieves higher accuracy levels by lowering the number of false positive incidents which standard IDS detection often produces.

Deshmukh and Ravulakollu [18] developed a CNN-based paradigm for an IoT-focused Intrusion Detection System (IDS), leveraging the spatial dependencies inherent in network traffic data. Their proposed architecture enhances the identification of IoT cyber threats by effectively modeling intricate traffic patterns. The study demonstrates that CNNs are especially suitable for IDS in resource-constrained environments, such as IoT devices, where computational efficiency is critical. These findings highlight the growing potential of deep learning in fortifying cybersecurity across IoT systems. The architecture of the proposed enhanced CNN is illustrated in Figure 2.5.

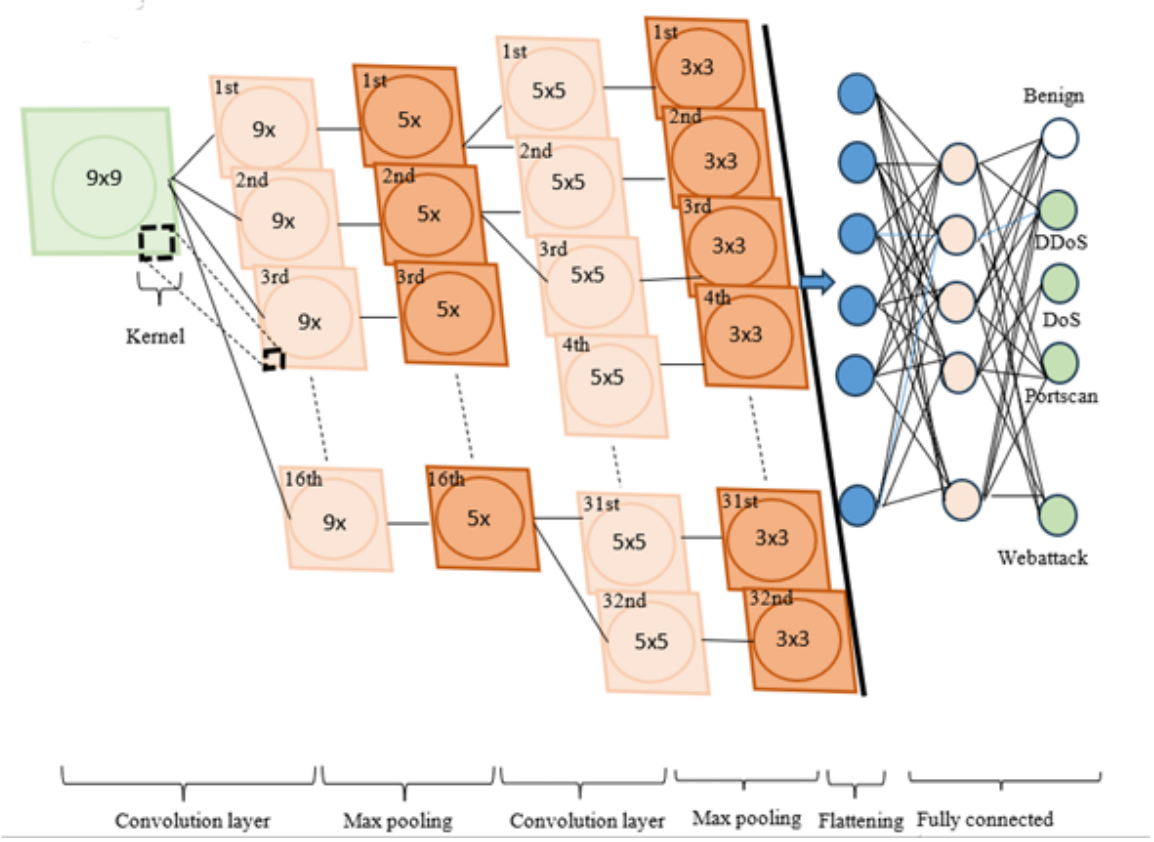


Figure 2.5: Enhanced CNN Architecture [18]

2.6.4 Ensemble Deep Learning Approaches

As described by Alalwany [13], MLP, CNN, LSTM, and GRU were combined into a stacking ensemble deep learning model to achieve more effective real-time intrusion detection in the Internet of Medical Things (IoMT) systems. In this context, CNN captures the spatial features and both LSTM and GRU capture the sequential dependencies. An MLP model enables finding relationships between different features. The effectiveness of the system against some types of ARP spoofing, DoS, Smurf, and Port Scan attacks is improved by MLP. Their results suggest that this approach benefits substantially from the ensemble systems architecture with regard to classifying intrusions in dynamic IoMT networks, both in single and multi-attack situations.

Gul [9] proposed the WGAN-DL-IDS model that combines Wasserstein Generative Adversarial Networks (WGANs) and deep learning with Random Forest classifiers to supply enhanced accuracy to intrusion detection systems. Their system utilizes WGANs for creating synthetic attack information to mitigate class imbalance as well as enhancing generalized performance of the model. After that, Random Forest method is applied for the feature selection technique to minimize dataset dimensionality before classification. Therefore, intrusion detection depends on models (MLP, CNN, and LSTM), which are

good at pattern and sequence data learning.

Li [12] implemented a novel model called ADFCNN-BiLSTM, which integrates a Fully Attention-Driven Convolutional Neural Network (Fully ADFCNN) with a Bidirectional Long Short-Term Memory (BiLSTM) network for intrusion detection systems. The use of deformable convolutional layers in the ADFCNN component enhances spatial feature extraction, allowing the model to better adapt to fluctuations in network traffic patterns. At the same time, the BiLSTM component captures long-range sequential dependencies, improving the system's ability to detect temporal attack patterns. This architecture exploration of attention mechanisms improves the feature representation, thus this is advantageous in detecting zero-day threats. The structure of the BiLSTM module used in this approach is depicted in Figure 2.6.

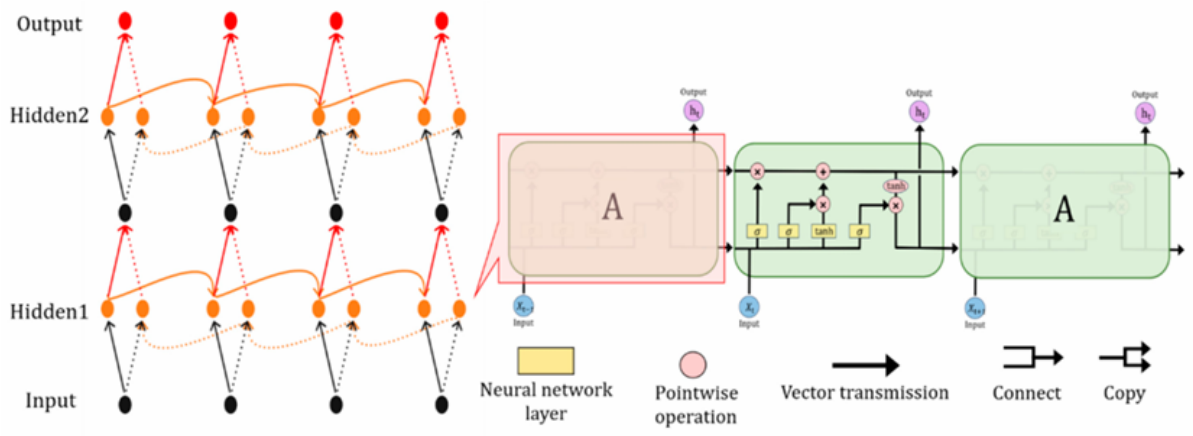


Figure 2.6: BiLSTM module [12]

2.6.5 GNN Approaches

Tran and Park developed the FN-GNN model in [11], which applies graph embedding to augment the performance of Graph Neural Networks (GNNs) for Intrusion Detection Systems (IDS). In this approach, each network's information pattern is modeled using graph data structures to recognize sophisticated cyber-attacks by analyzing relationships in traffic activity. Their method enhances feature extraction during GNN operations and significantly improves the detection of novel attack types. As shown in Figure 2.7, the proposed model architecture demonstrates how graph embedding is integrated into the GNN to strengthen its detection capabilities. Their findings support the viability of GNNs as a powerful machine learning approach for next-generation IDS.

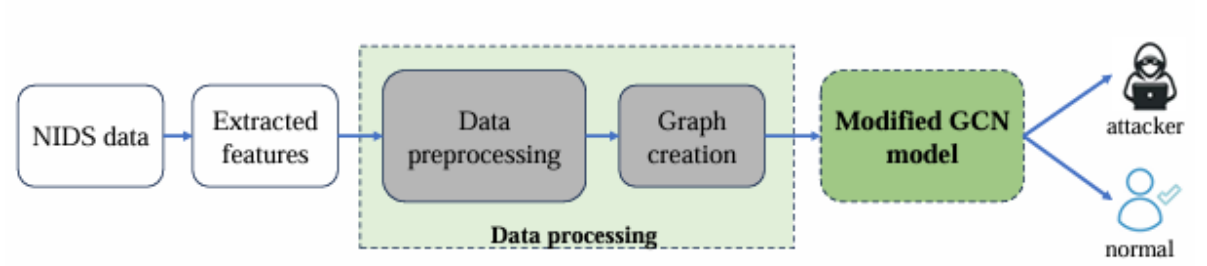


Figure 2.7: The proposed model [11]

Le and Park [19] improved the classification accuracy and model stability of Graph Neural Network (GNN)-based Intrusion Detection Systems (IDS) through feature rearrangement techniques. Their method restructures the input features in GNNs, so that benign and hostile actions can be differentiated more easily in complex network settings. Through their method, researchers manage to address overfitting along with improving generalization by enhancing feature representation. Their work has shown that GNNs are very useful within IDS as they were able to detect multi-class attacks with more accuracy. In particular, their evaluation demonstrated improvements in recall and precision across minority classes, which are often underrepresented in IDS datasets. This contribution highlights the potential of input feature engineering in boosting GNN model performance for cybersecurity applications.

Zhang [10] analyzed most of the advancements made in the application of deep learning for the Intrusion Detection System (IDS), as well as the primary difficulties like data imbalance and spatio-temporal feature extraction. They show that simple deep learning approaches find it challenging to work with imbalanced assets and model both space and time traits in network traffic. To provide a solution, they sought out newer models, features of attention-based architectures and GNNs, which provide greater representational flexibility and feature accommodation. The findings indicate that the proposed novel deep learning approaches would increase the efficiency and strength of IDS solutions. Moreover, the review emphasized the value of integrating generative models and transfer learning to overcome the scarcity of labeled intrusion data. Their work serves as a roadmap for future IDS research, especially in high-dimensional and real-time network environments.

Chapter 3

Methodology

This chapter gives an overview of the entire methodology adopted in building and evaluating machine learning models for network intrusion detection. The chapter begins with introducing the benchmark datasets utilized in this research and their corresponding characteristics and suitability for intrusion detection tasks. Subsequent to the data preparation process, the various deep learning architectures that were deployed and experimented with like 2D CNN, MLP, TabNet and CNN-BiLSTM model are elaborated. Finally, the chapter concludes by defining the standard performance metrics used to compare the models built systematically and comparatively.

3.1 Benchmark Datasets

CIC-IDS2017 is a usual benchmark data for intrusion detection systems, which provides five days of simulated traffic for normal behavior and many types of cyber attacks gathering. The dataset originally had 3,119,345 samples and 83 extracted features but were reduced to 2,830,540 samples following the removal of 288,602 unlabeled instances and 203 missing instances [9]. The distribution of attacks, shown in Figure 3.1, highlights the dataset's consideration of diverse attacks such as DoS, DDoS, web attacks, penetration attempts, brute force SSH attacks, Heartbleed, XSS, SQL Injection, Infiltration, Port Scans, and botnet activity. With its new threat categories, and formal tagging, CIC-IDS2017 provides a critical foundation for testing and training modern intrusion detection systems.

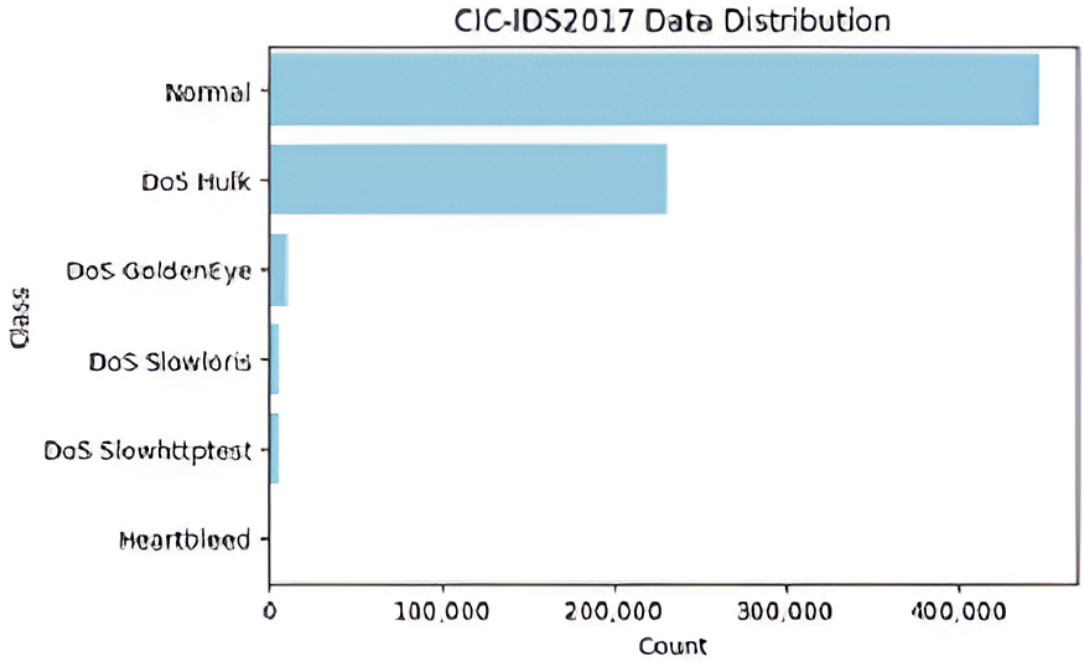


Figure 3.1: CIC-IDS2017 Data Distribution [7]

The IDS evaluation benchmark known as UNSW-NB15 was established by the Cyber Range Lab at the Australian Centre for Cyber Security (ACCS). It was generated using the IXIA Perfect Storm tool to simulate both real-world normal network traffic and contemporary attack behaviors in a controlled environment [12, 11]. The dataset contains 2,515,798 network flows 2,218,761 benign and 321,283 malicious organized into ten categories: fuzzers, analysis, backdoors, DoS, exploits, generic, reconnaissance, shellcode, worms, and benign traffic [19, 11]. Figure 3.2 provides a clear breakdown of this class distribution, emphasizing the dataset's diversity and value for evaluating intrusion detection techniques [12].

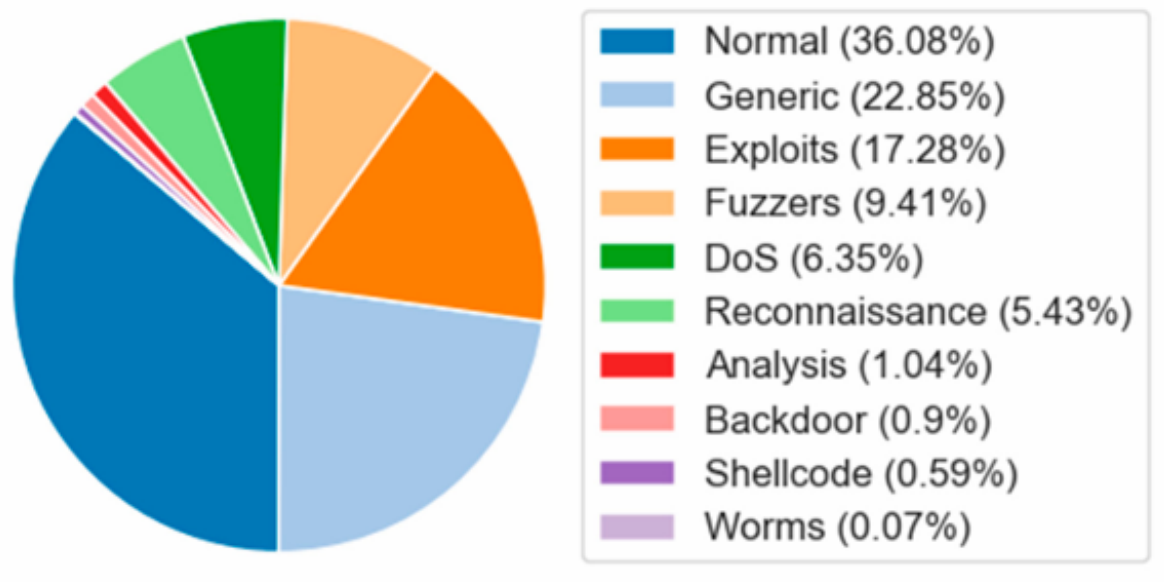


Figure 3.2: UNSW-NB15 Data Distribution [12]

The CSE-CIC-IDS2018 dataset created by CIC and CSE serves as a benchmark for intrusion detection systems (IDS) by storing real-world network traffic integrated with normal operations and a multitude of cyber-attacks such as Brute Force, DoS, Heartbleed, Web Attacks, Infiltration, Botnet, and DDoS [6]. It is obtained with the aid of CICFlowMeter which captures varying temporal information such as IP addresses, Ports, Protocols, packet sizes, and ATTACK labels, giving a reasonable feature set for analysis. Because of the class imbalance and redundancy of features, methodologies such as feature selection and missing value assignment are mandatory. This dataset functions as a preferred choice to test machine learning-based IDS models because it contains diverse security network scenarios.

3.2 Data Preprocessing

Preprocessing was the necessary process employed to clean the UNSW-NB15 dataset for model training. Although the UNSW-NB15 dataset contained many network flow features, the occurrence of issues such as class imbalance, missing records, duplicates, outliers, and skewed distributions were typical. Undoing such losses assists in polishing up the quality of data, allowing the learning algorithms to generalize better and converge toward higher accuracy. The following below are some of the major preprocessing techniques applied.

3.2.1 Handling Duplicates and Missing Values

The first thing to do was to clean the dataset by deleting duplicated rows with no redundancy of data, and this would skew the learning process. Secondly, the target column attack_cat had missing values, which were replaced with 'normal' using the assumption that unlabeled records are benign. The column was also normalized while removing the whitespace and changing all records to lowercase in order to make it uniform. Certain columns such as ct_flw_http_mthd and is_ftp_login contained missing values particular to the domain. Missing values in ct_flw_http_mthd very likely indicated that there were no HTTP methods available and therefore zeros were imputed. Missing values in is_ftp_login were also imputed with 0, taking them as instances where the user did not log in via FTP.

3.2.2 Outlier Detection and Treatment

Outliers can skew the statistical distributions, and they compromise performance of the model. To address this, all numeric columns-excluding time-related and identifier columns were processed using the Interquartile Range (IQR) method. For each selected numeric feature, values lying below the lower bound or above the upper bound were considered outliers and were replaced with the median of the respective column. The method retains the central tendency and minimizes the role of extreme values.

3.2.3 Skewness Correction

Many features in the dataset were strongly skewed, which makes machine learning models weak, especially those with assumptions of normality. In coping with this issue, a skewness-driven transformation was applied: positively skewed features were natural log-transformed while negatively skewed ones were log-transformed with regards to distance to their peak. A small amount of epsilon was added to avoid log(0) errors. This normalization moved distributions towards Gaussian-like shape and spurred learning efficiency.

3.2.4 Feature Engineering

Feature engineering was used to enrich the dataset with new informative features. For example, a new duration feature was created by computing Stime (start time) - Ltime (end time). Some ratio-based features such as byte_ratio, pkt_ratio, and load_ratio were created to represent the ratio of source and destination features. Interaction terms such as products of packets and bytes were also included to detect complex relationships between features. Measures of aggregated properties (total_bytes and total_pkts) furnished additional information which could not be quickly ascertained from individual values.

3.2.5 Removal of Highly Correlated Features

Highly correlated features bring in redundancy and the possibility of overfitting. A correlation matrix was calculated for all numerical features, and features that had a correlation

coefficient of 0.75 or more were indicated. From every highly correlated pair, only one feature was kept while the other one was discarded to minimize multi-collinearity. From this process, a minimal set of features was obtained with very little loss of information. Feature correlation analysis, as shown in Figure 3.3, determines strong correlation pairs, from which redundant attributes can be removed.

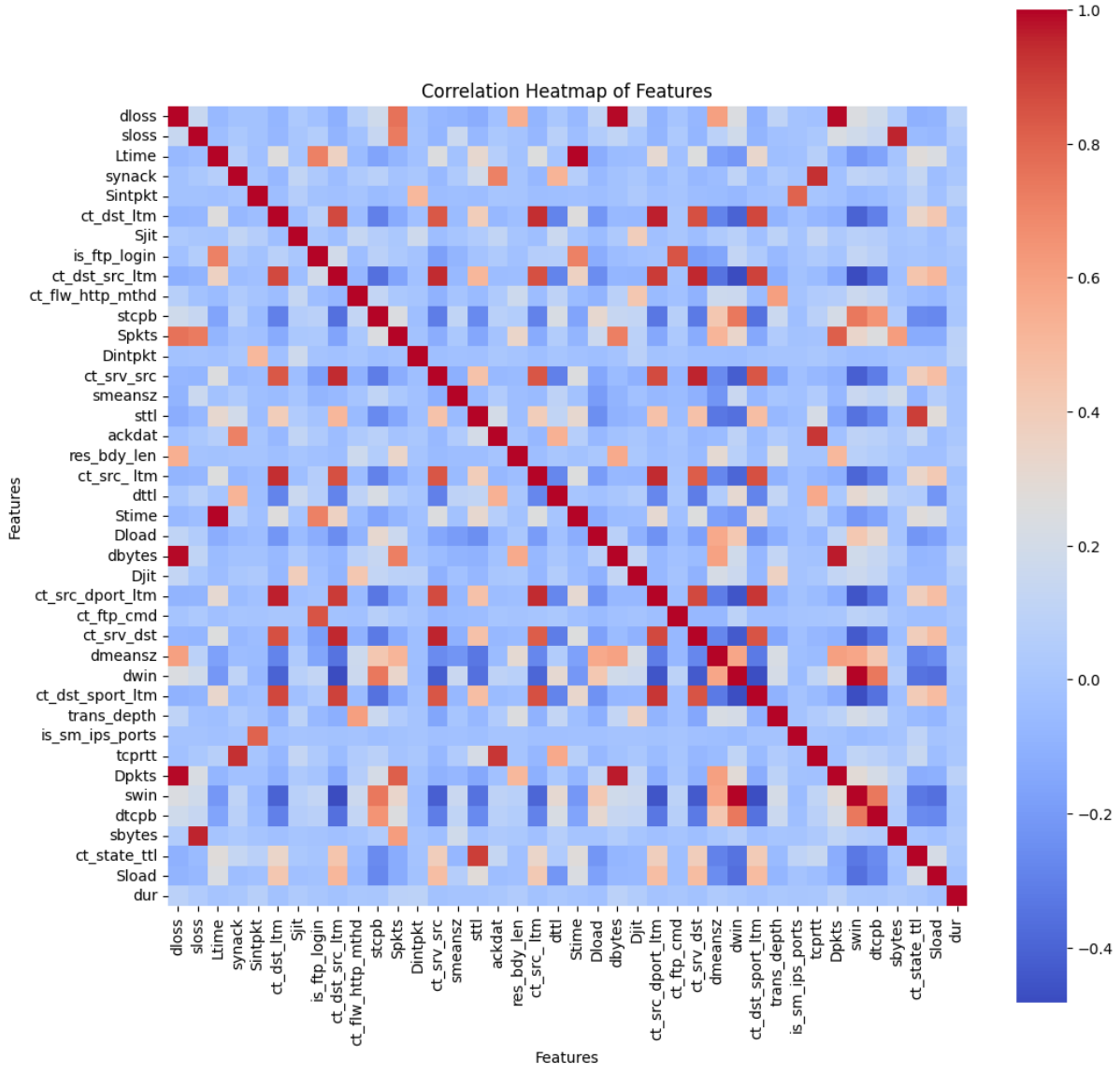
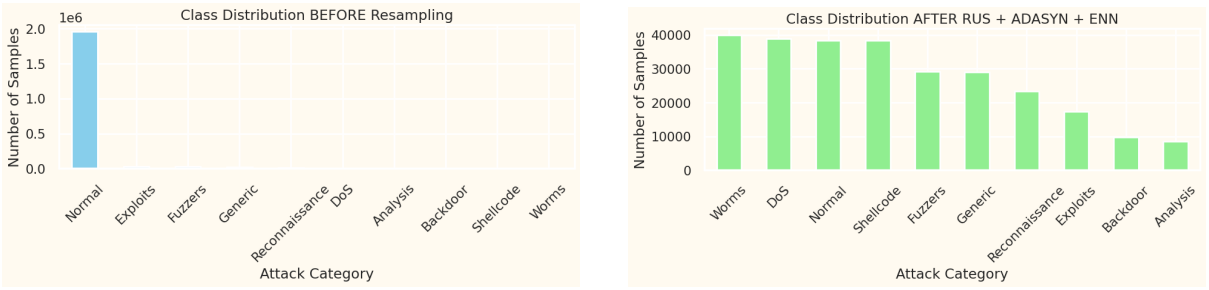


Figure 3.3: Corrleation graph between features

3.2.6 Addressing Class Imbalance

The original dataset had very high class imbalance as can be seen in the figure 3.4a, with the 'Normal' class being very highly represented with several attack categories being very underrepresented. To address this, a sequential resampling pipeline was employed.

First, RandomUnderSampler (RUS) reduced the dominant 'Normal' class. Then, Adaptive Synthetic Sampling (ADASYN) adaptively oversampled the minority attack classes by generating synthetic samples, focusing on harder-to-learn instances. Finally, EditedNearestNeighbours (ENN) cleaned the resulting dataset by removing potentially noisy samples near class boundaries. This combined RUS → ADASYN → ENN approach aimed to create a more balanced dataset, facilitating more robust and equitable model training across all classes. The resulting class distribution after this balancing process is shown in Figure 3.4a. While not perfectly uniform due to the adaptive nature of ADASYN and the cleaning effect of ENN, the distribution is significantly more balanced than the original.



(a) BEFORE Resampling

(b) AFTER Resampling

Figure 3.4: Comparison of Multi-Class Distribution Before and After Resampling.

3.2.7 Feature Selection

After balancing, mutual information was employed to evaluate the relevance of each feature with respect to the target variable. Features scoring below 0.01 were deemed irrelevant and subsequently removed. This feature selection step improved computational efficiency, reduced the dimensionality of the input space, and retained only the most informative features. As previous studies have shown, feature selection is essential for optimising intrusion detection system (IDS) performance by streamlining data processing and enhancing classification accuracy [20]. Building on this, our approach leveraged mutual information within an integrated selection strategy to achieve both high detection accuracy and computational scalability. The resulting feature relevance scores are visualized in Figure 3.5, which highlights the most influential features contributing to the classification task.

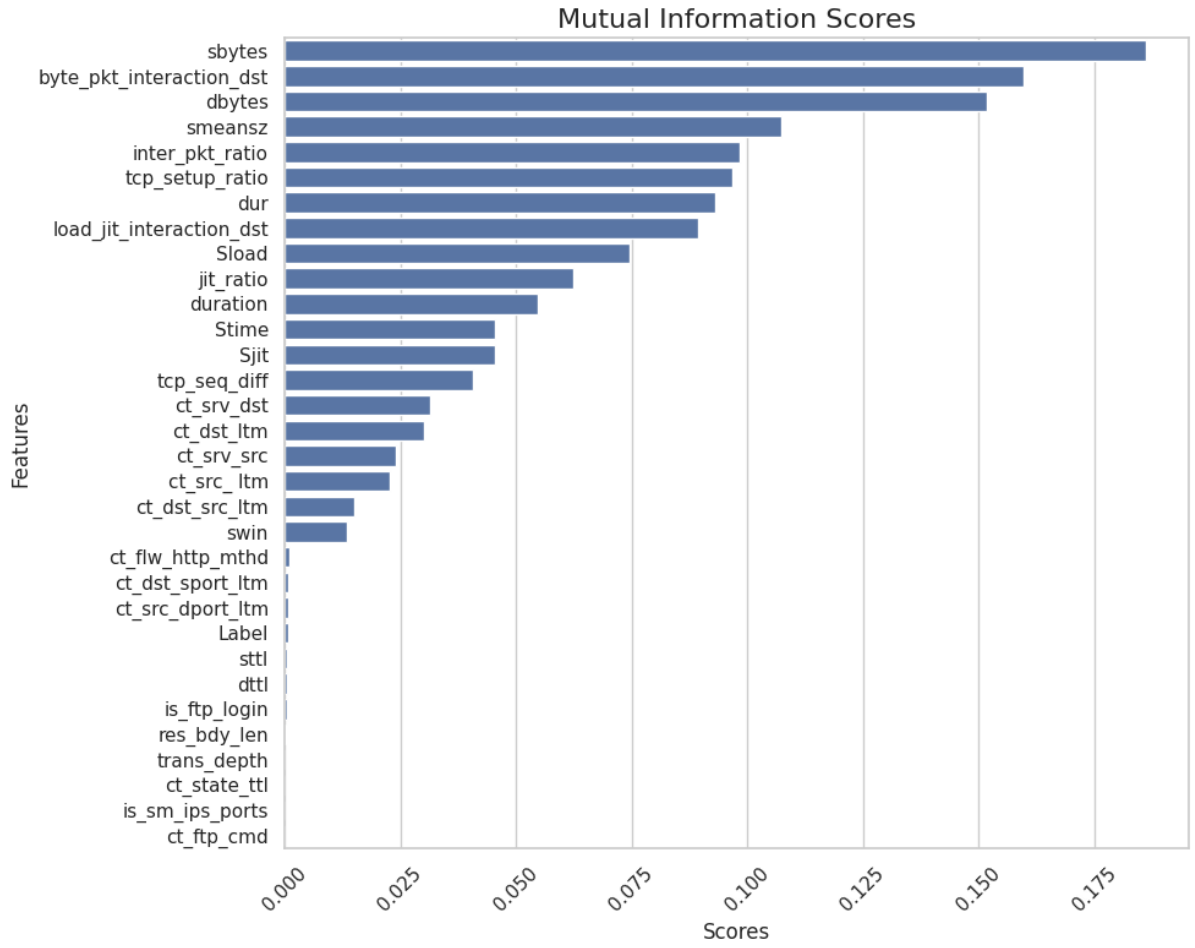


Figure 3.5: Mutual information of each feature

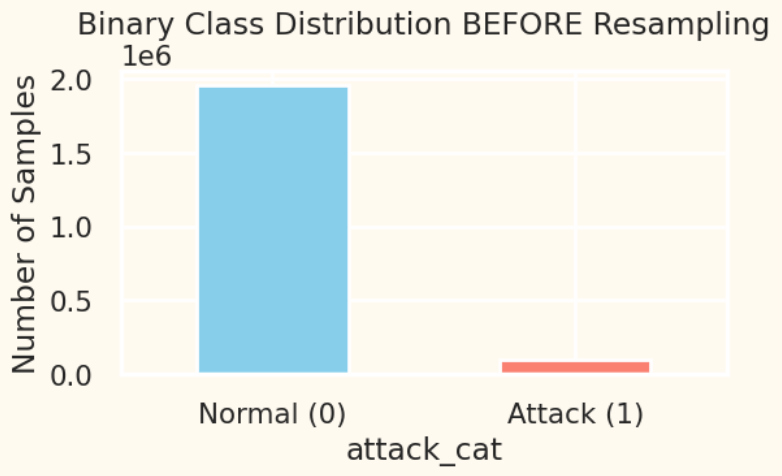
3.2.8 Feature Scaling

In order to ensure that all features had an equal contribution to the learning process of the model, standardization was done using StandardScaler. The method mapped all features to have zero mean and unit variance, which is particularly necessary for distance-based algorithms and gradient descent optimization.

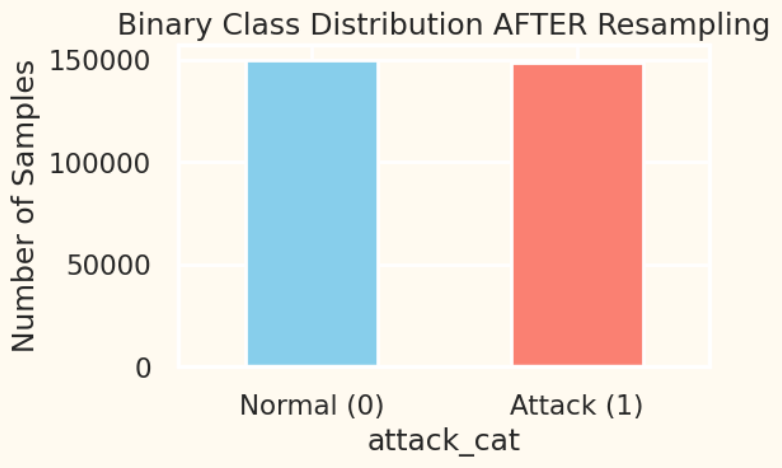
3.2.9 Binary Preprocessing

For the binary classification task (Normal vs. Attack), the features underwent the same initial preprocessing as for the multi-class scenario. The target labels were then converted to a binary format, designating the 'Normal' class as 0 and collapsing all attack classes into a single 'Attack' class labeled as 1. To address the inherent class imbalance in this binary setup, as illustrated in Figure 3.6a, a combined resampling approach was employed, utilizing ADASYN for oversampling the minority (Attack) class and RandomUnderSampler for undersampling the majority (Normal) class. This strategy resulted

in a balanced dataset specifically for binary model training, containing a target of 150,000 samples for each of the Normal and Attack classes, shown in Figure 3.6.



(a) BEFORE Resampling



(b) AFTER Resampling

Figure 3.6: Comparison of Binary Class Distribution Before and After Resampling

3.3 Proposed Approach

In this study, the UNSW-NB15 dataset was used to evaluate the performance of various deep learning models for intrusion detection. Its diverse attack types and realistic network traffic, generated using contemporary tools and attack behaviors, make it ideal for training and testing robust detection models capable of handling modern cyber threats. The dataset's comprehensive feature set, capturing essential aspects of network flows, provides a rich foundation for developing and assessing sophisticated machine learning approaches. The overall methodology employed in this research, which systematically encompasses initial data acquisition, extensive preprocessing tailored for both multi-class

and binary scenarios, the detailed design and implementation of four distinct deep learning architectures, and their rigorous evaluation, is outlined in Figure 3.7. This structured approach ensures a thorough and comparative assessment of each model's capabilities and limitations within the context of network intrusion detection.

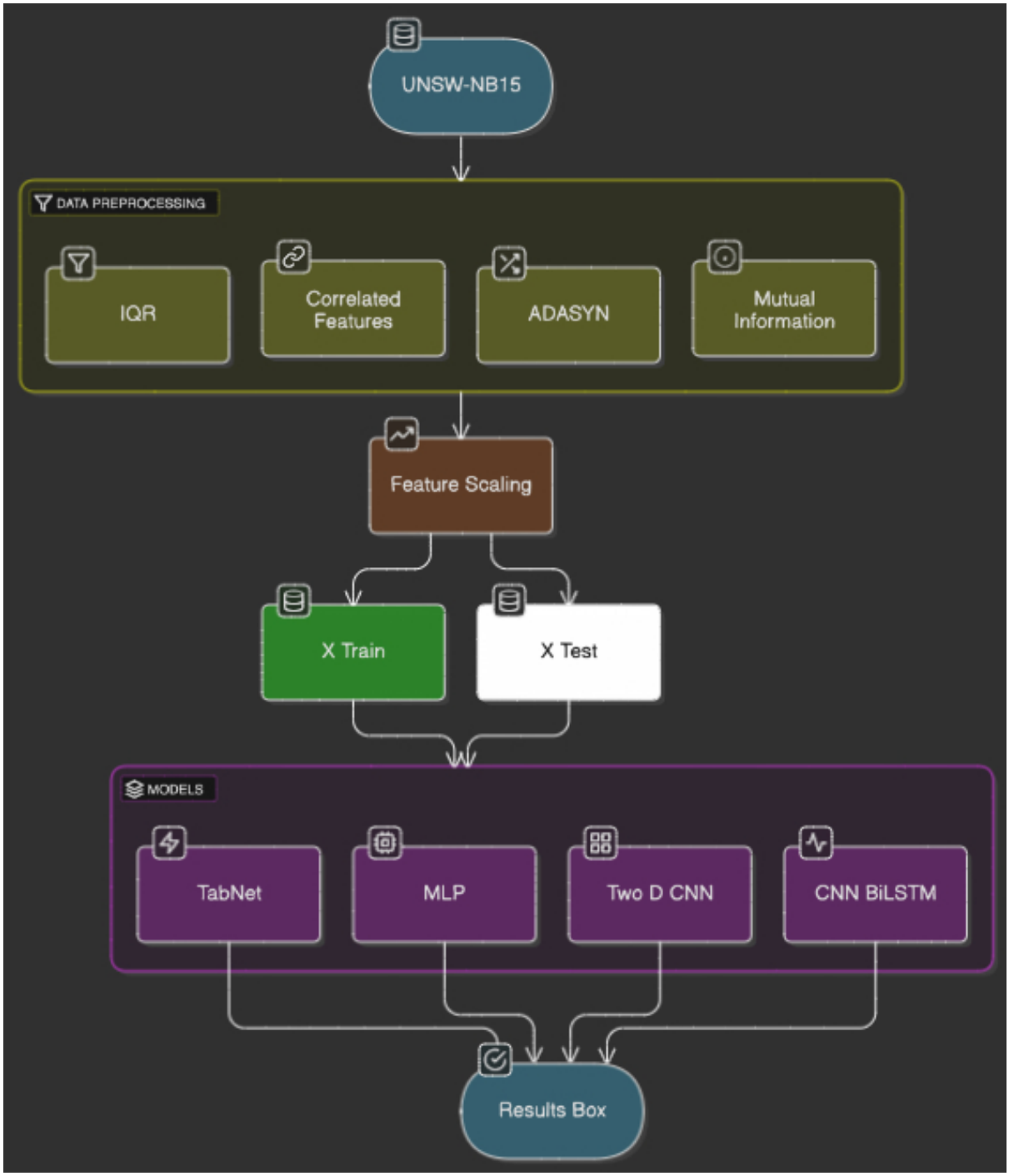


Figure 3.7: Flowchart of the Proposed Methodology

3.3.1 2D Convolutional Neural Network (2D CNN)

Among the various architectures tested, the 2D Convolutional Neural Network (2D CNN) model was the first model implemented and evaluated. This architecture is particularly effective at extracting spatial patterns from two-dimensional representations of network traffic data. It was chosen as a baseline deep learning model due to its ability to capture local feature correlations, which are often indicative of different intrusion behaviors. Its relatively straightforward structure and efficient computation make it a practical choice for early experimentation in model selection.

The model was implemented using TensorFlow's Keras API in a sequential format. The architecture begins with a Conv2D layer consisting of 32 filters with a kernel size of (3×3), using ReLU activation and 'same' padding to retain the original spatial dimensions of the input. This layer is designed to scan for local spatial patterns within the input data. The extracted features are then downsampled using a MaxPooling2D layer with a pool size of (2×2), which reduces the computational complexity and helps mitigate overfitting. A Dropout layer with a rate of 0.3 follows to further prevent overfitting by randomly deactivating a subset of neurons during training. This initial sequence of layers is critical for compressing input complexity while retaining key discriminative features.

There is followed a second Conv2D with 64 filters, with the same (3×3) kernel size and then another MaxPooling2D and Dropout layer with dropout rate 0.3. Convolutional stacks play the role of giving the model the capacity to learn ever more abstract form of the data. An output from the preceding pooling layer is flattened to become a 1D vector by the Flatten layer and is then sent into the Dense layer that has 128 neurons and ReLU activation, in order to extract the high level features. A last Dropout for 0.4 is added after the last layer. This layered design helps the model transition smoothly from low-level pattern extraction to high-level classification reasoning.

The model concludes with a Dense output layer with softmax activation in which the number of neurons is equal to that number of the unique attacks categories in the dataset. This layer has the following output; multi-class classification class probabilities. The model was assembled with the Adam optimizer along with a learning rate of 0.001 and applied the categorical crossentropy loss function, which is appropriate for using on multi-class classification issues. Two indices of performance were calculated – both accuracy and AUC (Area Under the Curve) score were evaluated. These metrics provide a comprehensive assessment of the model's overall classification strength and its robustness in distinguishing between normal and attack traffic.

Training was carried out over a maximum of 100 epochs with a batch size of 32. Two callback strategies were used to ensure efficient training: EarlyStopping, with a patience of 10 epochs to stop training when validation loss ceased to improve, and ReduceLROnPlateau, which automatically reduced the learning rate by a factor of 0.5 if no improvement in validation loss was observed over 5 epochs, with a lower bound of 1e-6. A validation split of 20 percent was reserved from the training data to evaluate generalization performance during training. Final performance was assessed on the test set using a detailed classification report, including precision, recall, and F1-score for each class. The architecture layout used in this implementation can be found in Figure 3.8, summariz-

ing the structure and flow of the 2D CNN model. This model served as a foundational benchmark against which the more complex architectures were later compared.

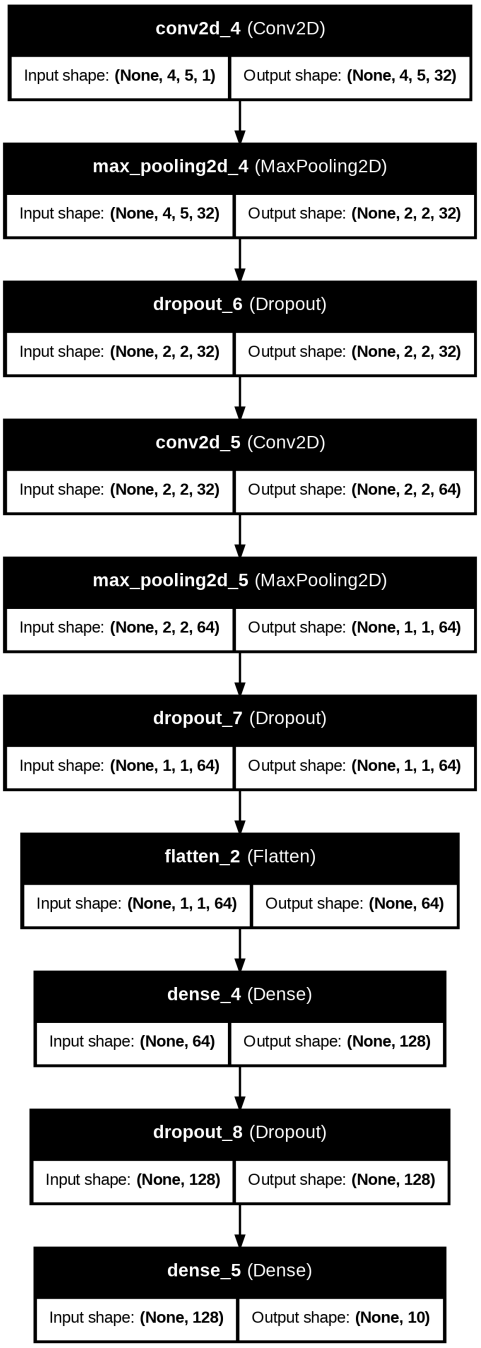


Figure 3.8: 2D-CNN Model

3.3.2 Multi-Layer Perceptron (MLP)

The second model experimented in this paper is a Multi-Layer Perceptron (MLP) with a series of fully connected layers used to create complex, non-linear relationships against structured high-dimensional tabular data. MLP works extremely well wherever features are already preprocessed and normalized, i.e., when working with a structured UNSW-NB15 dataset employed in this paper. Its ability to model intricate decision boundaries makes it a compelling choice for intrusion detection tasks. Moreover, its relatively simpler architecture compared to recurrent or convolutional networks makes it efficient and easy to deploy.

The model was developed and implemented using TensorFlow's Keras API under a sequential structure. Preprocessing of the dataset was done before training with an 80/20 stratified split to ensure class distribution across training and test sets. Labeling was performed using one-hot encoding to match the categorical format of output required by softmax activation in the output layer. This encoding scheme transforms categorical class labels into a binary matrix, facilitating compatibility with the softmax layer. Such preprocessing steps ensure the model is both well-calibrated and equipped to handle imbalanced data.

The architecture begins with a 256 neuron-dense input layer activated with ReLU neurons, then applies a Batch Normalization layer for learning regularization and the acceleration of convergence. The use of a dropout layer at 0.4 kept them from overfitting by switching off neurons, randomly while training. This was then followed by a second dense layer of 128 neurons and dropout of 0.3, and a third dense layer of 64 neurons with dropout of 0.2, each with batch normalization to ensure stable propagation of gradients. These layers are designed to progressively extract and refine complex patterns from the input feature space. The use of dropout and batch normalization together forms a balanced approach to improving both model generalization and convergence speed.

The output layer included 10 neurons, one for a different class in the UNSW-NB15 dataset. Output probability distributions over classes were softmax-activated. The model was compiled using the Adam optimizer with a learning rate of 0.001, and the loss function selected was categorical crossentropy, appropriate for multi-class classification tasks. Performance and precision of the models and AUC were observed during training. These evaluation metrics offer insight into both the general classification quality and the model's ability to discriminate between classes. Such a setup is particularly important for intrusion detection systems where false positives and false negatives can have serious implications.

A maximum of 100 epochs with batch size 32 was used for training. Early stopping with patience 10 was used to prevent overfitting, and the recovery of the best weights by loss on validation when it stopped to progress. At last, the model was used to predict the test set and classification report containing precision, recall, and F1-score for each class was generated after training. The architecture of the model is shown in Figure 3.9 as a drawing of the MLP. The use of early stopping ensured computational efficiency while still preserving optimal performance. This model provides a strong baseline to compare against more complex architectures used in this study.

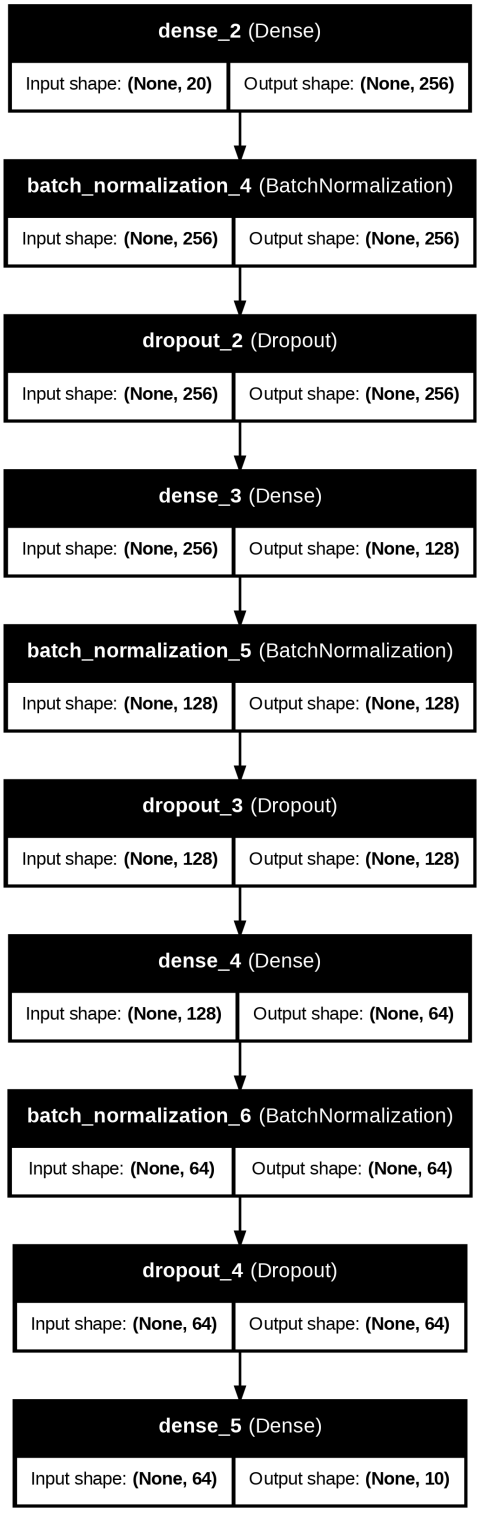


Figure 3.9: MLP Model

3.3.3 TabNet

The third model applied in this work is TabNet, which is a custom-developed deep learning model specifically designed for tabular data. Unlike regular neural networks, which treat tabular features as equal to image or text features, TabNet applies sequential attention to determine the most relevant features at each stage of the decision-making process to enable interpretability and tabular dataset-efficient learning. Its intrinsic feature selection ability and ability to model complex relationships without further preprocessing are causing it to be an appealing choice for structured intrusion detection tasks.

In this implementation, the model was trained on the preprocessed UNSW-NB15 dataset. Labels were first one-hot encoded and later converted back to integer format using argmax to meet TabNet's input requirements. The input features were standardized to float32 to match TabNet's expected data type, and a stratified train-test split (80/20) was applied to ensure balanced class representation.

The TabNet classifier was initialized with 64 decision (n_d), and attention (n_a) dimensions, and 5 decision steps (n_steps), thus letting it learn complex feature interactions at several stages. In order to improve generalization, the model used n_independent=2 and n_shared=2, where the first defines number of independent and second defines a number of shared layers per decision steps. A gamma of 1.5 was applied to regularize reusing the features between steps leading to model attention to specific subsets of the features at each step of the prediction.

Training was implemented on the Adam optimizer with a learning rate of 0.02 and a StepLR learning rate scheduler that would decay the learning rate by 0.9 after every 10 epochs. The training did not stop after a maximum number of 100 epochs (with early stopping (patience = 10) in place which prevented the overfitting of the data. A batch size of 1024 and a virtual batch size of 128 were applied to take advantage of memory use and training speed.

andard classification metrics were reserved for test set, and model evaluation was carried out on the held-out test set. The accuracy, precision, recall and F1-score were presented in the classification report produced, allowing a broad understanding of how the model performed during all attack categories.

3.3.4 CNN-BiLSTM

One of the several tested architectures preferred the CNN-BiLSTM model because of its best classification performance. This combination of Convolutional Neural Networks (CNNs)' spatial feature extraction power and bidirectional long short term memory (BiLSTM) networks' sequential learning conditions lets it take advantage of its spatial and temporal relations in the network data. This synergy makes the model particularly effective in capturing complex attack patterns across multiple dimensions of network behavior. It provides a robust framework capable of modeling both localized packet features and long-term dependencies in traffic flow.

TensorFlow's Keras API was used in a sequential design to implement the model. It starts with a Conv1D in one dimension having 64 filters, with a kernel size of 3, that

detects local patterns of features in the input data. There is then a MaxPooling1D layer that reduces the spatial size and reduces the overfitting. In order to increase training stability and speed, Batch Normalization is used following pooling. These layers help form a strong foundational representation before passing data into the temporal learning phase. This early structure is essential for reducing computational complexity and improving convergence during training.

Next, the output goes through bidirectional LSTM with 64 units returning contextual information from both forward and backward directions. A second step of MaxPooling and Batch Norm is taken, followed by a deeper Bidirectional LSTM with 128 units. This layer produces a fixed output vector of dimension length, it is then sent to a Dropout layer with the rate of 0.6 in order to avoid over fitting. The stacked LSTM layers allow the model to learn long-range dependencies crucial for distinguishing between subtle attack signatures. Dropout further ensures that the model does not become overly reliant on specific pathways, thus enhancing generalization.

This final classification is done by a Dense output layer (10 neurons) in which each neuron is related to a particular attack category in the UNSW-NB15 dataset. A softmax activation function is employed in order to generate a probability distribution over the output classes. The model is compiled with the Adam optimizer, learning rate of 0.001 and categorical crossentropy loss, a good fit for multi-class classification problems. During training, model performance was observed using both accuracy and Area Under the Curve (AUC) metric. These metrics provided a balanced evaluation of the model's overall performance and its capability to handle imbalanced class distributions. The choice of softmax activation ensures that the model outputs interpretable probability scores across all classes.

The training process had a maximum epoch of 100 and batch size of 32. Early stopping with patience of 10 was used to stop training after the validation loss was no longer getting better thereby preventing over fitting and saving the best weights. A validation split of 20 percent of training data was held out to monitor generalisation while training. When finished, performance of the models was measured on the test set with a detailed classification report that included a precision, recall and F-measure for each class. Architecture of the CNN-BiLSTM model is presented in Figure 3.10. This comprehensive setup ensured that the model remained both accurate and generalizable across unseen network traffic. It also allowed for efficient use of computational resources during training, as all experiments were executed using a GPU-enabled environment on Google Colab.

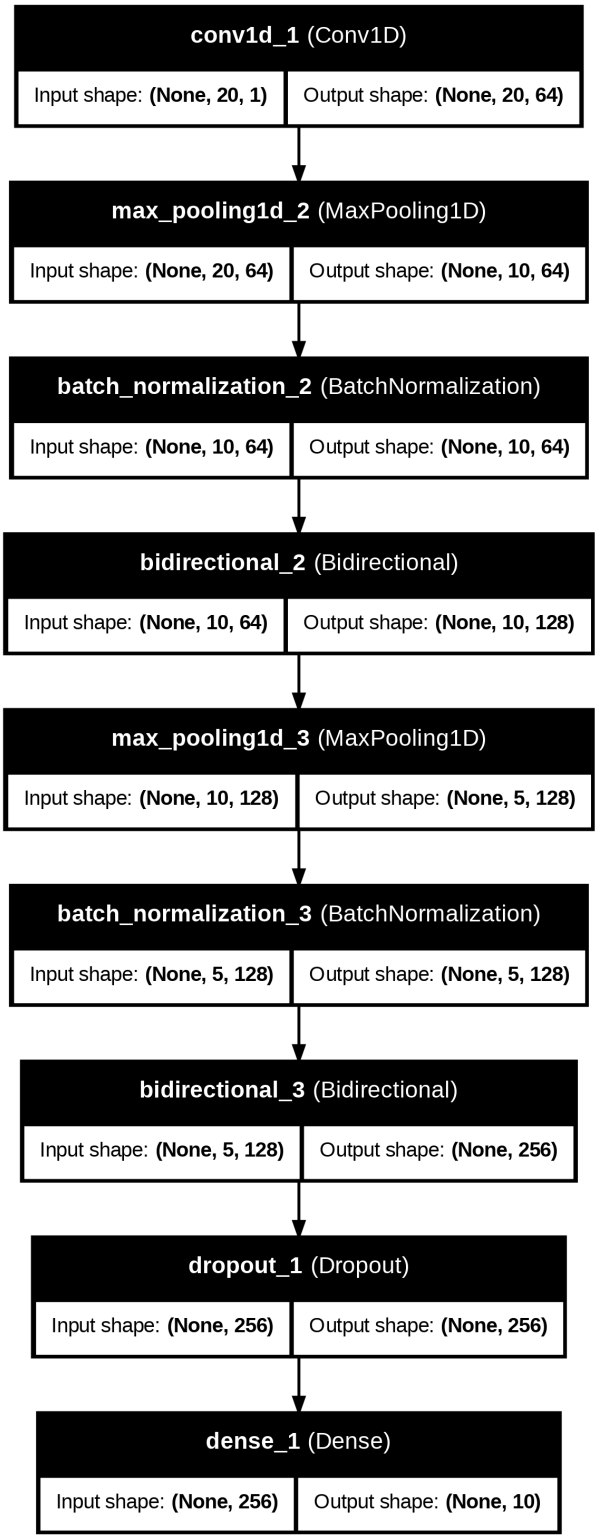


Figure 3.10: CNN-BiLSTM Model

3.4 Evaluation Metrics

In order to measure the performance of the model a number of classification metrics has been used such as accuracy, precision, recall and F1-score. Such metrics are crucial to evaluating the efficacy of classification models when the work with imbalanced datasets or has multi-class classification issues.

3.4.1 Accuracy

Accuracy determines the complete correctness of the model by computing the ratio of predicted correctly to the total number of predictions made. It is defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

where:

- TP: True Positives
- TN: True Negatives
- FP: False Positives
- FN: False Negatives

3.4.2 Precision

Precision concerns the relevance of the positive predictions. It is the ratio made up of correctly predicted positive observations to total predicted positives.:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.2}$$

where:

- TP: True Positives
- FP: False Positives

High precision means that we get better results as compared to irrelevant ones.

3.4.3 Recall

Recall (or sensitivity or true positive rate) describes how well a model can recognize the set of relevant cases among a set of cases. It is defined as:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.3}$$

where:

- TP: True Positives
- FN: False Negatives

A high recall means the model successfully captures most of the actual positive cases.

3.4.4 F1-score

F1-score is the harmonic mean between the precision and recall, that represents a single measure of balance between these two. It is useful in situation of inequality of class distribution:

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.4}$$

An F1-score reaches its best value at 1 (perfect precision and recall) and worst at 0.

3.4.5 Area Under the ROC Curve (AUC)

The Area under the Receiver Operating Characteristic Curve (AUC) is a commonly used measure to assess the performance of classification models most especially in cases where data have an imbalanced distribution. A worst case scenario for multi-class learning is the average of OvR AUCs, averaged often with class support (weighting). AUC represents the model's capacity to separate classes and the higher the values, the better the model. It is especially appropriate in Intrusion Detection Systems evaluation, where it is important to differentiate between an attack and normal traffic.

These metrics were obtained on the test set to assess the classification performance of the models for all categories of attack and normal traffic. They give a thorough view of the strengths and shortcomings of each model specifically, in multi-class classification of circumstances where the use of only accuracy can be a misleading representation of performance. For instance, in class imbalance data sets, a model could reach high accuracy by just predicting the majority class, whereas, not able to identify minority attack types. Therefore, analysis of precision, recall, F1-score and AUC serves as a measure that guarantees that the model is robust in all classes, specifically, in detection of rare but important attack categories. In order to build Intrusion Detection Systems that will work effectively and reliably under real-world network conditions, such holistic evaluation is paramount.

Chapter 4

Results

This chapter provides a thorough assessment of the results of experiments implemented on the DL models created in order to detect network intrusions, discussed in the previous chapter dedicated to methodology. It takes a systematic look at findings for the four unique architectures studied in terms of performance outcomes. It is the 2D Convolutional Neural Network (2D CNN), the Multi-Layer Perceptron (MLP), TabNet, and the hybrid CNN-BiLSTM model. For every architecture, the analysis dives deep into critical training dynamics, general classification performance metrics, and a fine-grained analysis of their efficacy at classifying certain types of attacks as compared with normal data using classification reports and confusion matrices for in-depth information. The chapter ends up with a comparative analysis synthesizing the findings to reveal relative strengths, weaknesses and practical trade-offs of each model in the light of UNSW-NB15 dataset.

4.1 Experimental Results

This part reports a thorough assessment of the results achieved for every deep learning model considered in the current research. The results of the performance and characteristics of the two-dimensional Convolutional Neural Network, MLP, TabNet, and the hybrid CNN-BiLSTM model will be discussed in a sequential order. For each architecture, the discussion will include some critical aspects such as the training process (epochs needed and Google Colab GPU computing time spent); overall classification performance metrics, based on the testing, and there will be an evaluation of the model's capability of categorizing specific attack categories apart from the normal traffic, as was evident in their confusion matrices and classification reports.

The classification encodings applied in this study are given in Table 4.1.

Table 4.1: Class Encodings Used in the Study

Encoding	Class Name
0	Analysis
1	Backdoor
2	DoS (Denial of Service)
3	Exploits
4	Fuzzers
5	Generic
6	Normal (No attack)
7	Reconnaissance
8	Shellcode
9	Worms

4.1.1 2D CNN Model Results

The first deep learning model the two dimensional Convolutional Neural Network (2D CNN) was the first one that was tested. The training was run on Google Colab GPU and the model converged after 100 epochs, approximately taking 32 minutes to finish the training process.

The 2D CNN achieved a test accuracy of 81.6%, with a macro-averaged F1-score of 0.778 and a weighted F1-score of 0.808. For the most part these metrics describe good performance level. The classification task included the ten classes specified in Table 4.1, namely the normal traffic and attacks. This detailed performance metrics for each class are listed in Table 4.2.

Table 4.2: 2D-CNN Classification Report

Index	Precision	Recall	F1-score	Support
0	0.82	0.78	0.80	1829
1	0.71	0.52	0.60	1803
2	0.60	0.35	0.44	4001
3	0.59	0.61	0.60	7448
4	0.86	0.86	0.86	3591
5	0.93	0.83	0.88	5641
6	0.98	0.98	0.98	8000
7	0.76	0.93	0.84	4876
8	0.82	0.87	0.85	7574
9	0.88	1.00	0.94	7982
Accuracy			0.82	52745
Macro Avg	0.80	0.77	0.78	52745
Weighted Avg	0.81	0.82	0.81	52745

Although the model exhibited good efficiency in the detection of some classes especially Normal, Worms, Generic, Fuzzers and Shellcode, the efficiency behind the detection of the other classes is considerably weak. For example, DoS class demonstrated a relatively low recall (0.35) and the lowest F1-score (0.44) thus suggesting that the model has not been precise at distinguishing the instances of this category. The Backdoor and Exploits classes also showed relatively lower F1-scores (0.60, 0.60). This is further confirmed by confusion matrix (Figure 4.1). For the DoS cases, a large number of them were mistakenly classified as Exploits and Shellcode. Backdoor samples were used to mix with Exploits and Reconnaissance most of the time. Sampling for analyses were often classified as Exploits.

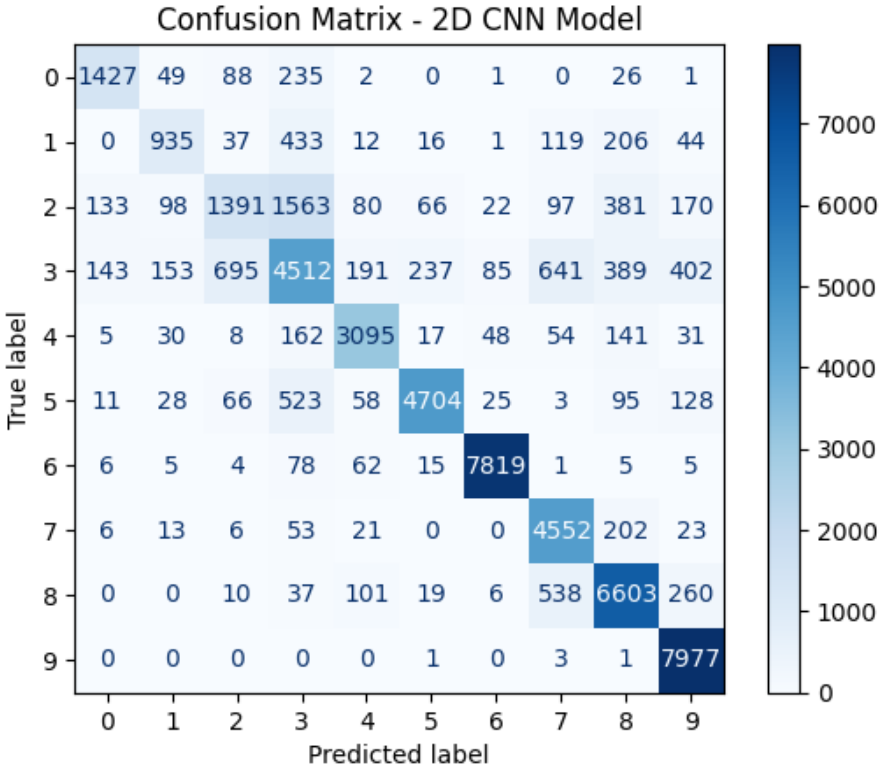


Figure 4.1: Confusion Matrix for the 2D-CNN Model

In conclusion, 2D CNN model demonstrated good overall accuracy and powerful ability to discern a number of unique traffic patterns, particularly Normal traffic and some of the attacks (such as Worms and Generic). Nonetheless, it had a great deal of difficulty in precisely classifying DoS attacks which it frequently mistook for Exploit or Shellcode. Difficulties also existed in distinguishing Backdoor and Analysis out of being mistakenly mislabelled as Exploits or Reconnaissance. Such results confirm that albeit 2D convolutional architectures are able to extract the relevant spatial information these results point out that much confusion still occurs between some of the attack classes, and, therefore, more advanced methods should be sought out to mitigate this confusion and disentangle

these complex patterns.

4.1.2 MLP Model Results

TThe second deep learning model which was assessed was the Multi-Layer Perceptron (MLP). Training was done using a Google Colab GPU, and after about 70 epochs, the model converged completing the training on the network in about 34 minutes.

The MLP achieved a test accuracy of 81.5%, with a macro-averaged F1-score of 0.781 and a weighted F1-score of 0.809. These measures represent strong overall performance in all of the ten types of classifications established in Table 4.1, including Normal traffic and several types of attacks. Detailed per-class metrics are shown in Table 4.3.

Table 4.3: MLP Classification Report

Index	Precision	Recall	F1-score	Support
0	0.87	0.76	0.81	1829
1	0.69	0.54	0.61	1803
2	0.59	0.38	0.46	4001
3	0.60	0.61	0.60	7448
4	0.87	0.88	0.87	3591
5	0.94	0.84	0.88	5641
6	0.99	0.98	0.98	8000
7	0.75	0.93	0.83	4876
8	0.82	0.84	0.83	7574
9	0.87	1.00	0.93	7982
Accuracy			0.82	52745
Macro Avg	0.80	0.78	0.78	52745
Weighted Avg	0.81	0.82	0.81	52745

Although the model proved a high efficacy in the detection of some of the attacks in specific, the performance was much weaker in others. For example, the recall value for DoS class was quite low (0.38), which led to the lowest F1-score (0.46) and thus, the model was not able to recognize the instances of this category with a high frequency. Performance on the Backdoor and the Exploits classes showed also a relatively poor performance, judging by their F1-scores (0.61 and 0.60, respectively). This is even more demonstrated by the revised confusion matrix (Figure 4.2). Importantly, the DoS samples were often mistaken to be a part of Exploits and Shellcode classes. The Backdoor samples were commonly mixed up with Exploits, Shellcode and Reconnaissance traffic. Also, Analysis samples were identified as Exploits and DoS in sometimes.

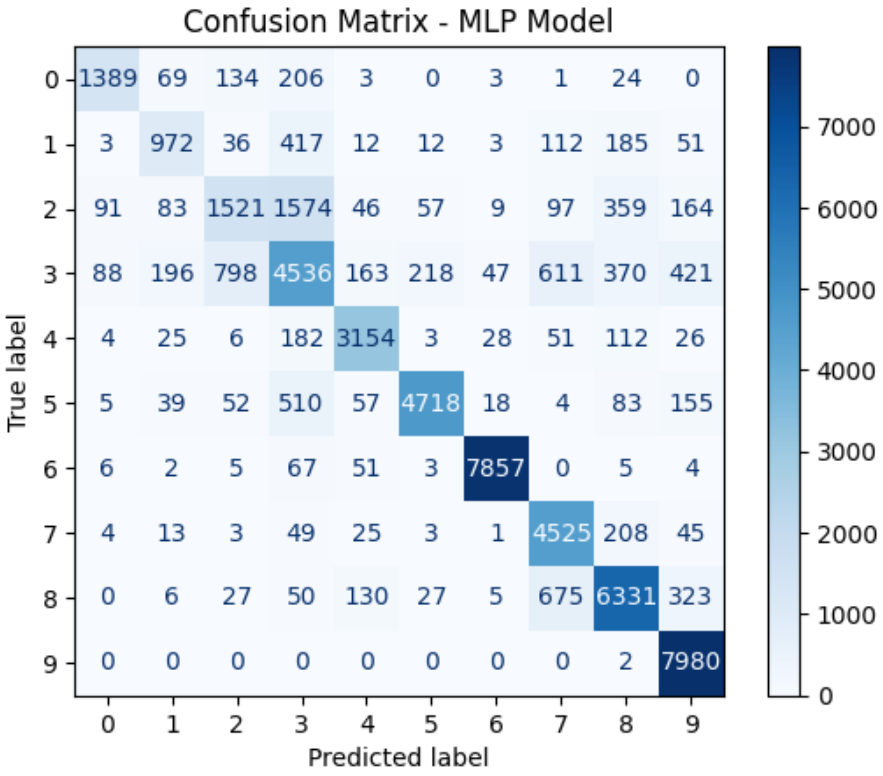


Figure 4.2: Confusion Matrix for the MLP Model

Overall, the updated MLP model showed good accuracy and capability in distinguishing several attack patterns, including excellent detection of Worms and Normal traffic. However, it struggled significantly with differentiating DoS attacks, misclassifying them often as Exploits or Shellcode. It also showed moderate difficulty with Backdoor and Exploits classes. These results suggest that while the MLP architecture improved its overall performance, it still faces difficulties separating classes with potentially overlapping features, particularly the challenging DoS category.

4.1.3 TabNet Model Results

The third model that was assessed was TabNet. A Google Colab GPU was used to conduct the training, and the model needed 100 epochs to train the model which took about 24 minutes.

TabNet achieved a test accuracy of 85.7%, with a macro-averaged F1-score of 0.831 and a weighted F1-score of 0.854. These are representative of its robust overall performance when it came to performing the classification task, which was on the ten categories introduced in Table 4.1. Considerably detailed performance metrics are represented in Table 4.4.

Table 4.4: TabNet Classification Report

Index	Precision	Recall	F1-score	Support
0	0.86	0.82	0.84	1829
1	0.76	0.64	0.69	1803
2	0.67	0.53	0.59	4001
3	0.66	0.67	0.66	7448
4	0.92	0.90	0.91	3591
5	0.95	0.89	0.92	5641
6	0.99	0.99	0.99	8000
7	0.79	0.91	0.85	4876
8	0.84	0.92	0.88	7574
9	0.96	1.00	0.98	7982
Accuracy			0.86	52745
Macro Avg	0.84	0.83	0.83	52745
Weighted Avg	0.86	0.86	0.85	52745

The TabNet model demonstrated strong classification accuracy for several attack categories, particularly Normal, Worms, Generic, and Fuzzers, indicating that it could effectively recognize distinct patterns within well-defined traffic. It also performed well on Shellcode, Exploits, Reconnaissance, and Analysis, showcasing its capacity to generalize across a broad spectrum of threats. This suggests that TabNet's attention mechanism is capable of prioritizing meaningful features in structured tabular data. However, its performance dropped notably for classes like DoS (F1-score: 0.59), Backdoor (0.69), and Exploits (0.66), highlighting challenges in identifying attacks with overlapping or subtle patterns. These results point to limitations in handling more ambiguous or noisy feature sets. Despite these challenges, the model overall maintained consistent performance across most attack vectors.

The updated confusion matrix (Figure 4.3) provides further insight into these shortcomings, illustrating key inter-class confusions. DoS samples were frequently misclassified as Exploits, Analysis, or Reconnaissance, indicating a lack of distinctive features for clear separation. Similarly, Exploits were often confused with DoS or Backdoor, likely due to overlapping behavioral signatures in their traffic patterns. Backdoor traffic was occasionally misidentified as Exploits or Reconnaissance, which may reflect shared traits in stealthy or lateral movement behaviors. Analysis samples were sometimes misclassified as Exploits, suggesting that network analysis tools can generate similar traffic signatures to malicious probing. These patterns highlight the need for either more discriminative features or model adjustments to better distinguish between these closely related attack types.

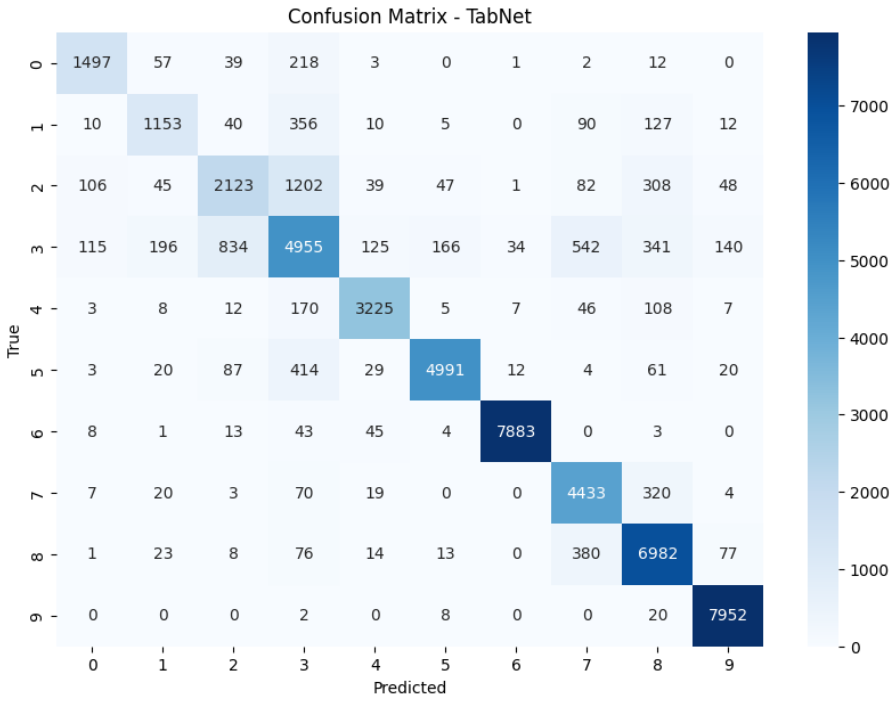


Figure 4.3: Confusion Matrix for the TabNet Model

Overall, the new version of TabNet model demonstrated strong accuracy for most of the classes and had high overall accuracy. It was good at recognizing separate pattern, such as Normal traffic, Worms and Generic attacks. Nevertheless, it still had major issues in robustly differentiating DoS attack and indicated moderate challenges in Backdoor and Exploits classes, meaning that there were remaining challenges in separating classes that may share overlapping or subtle features, despite attention mechanisms used by TabNet.

4.1.4 CNN-BiLSTM Model Results

The final model to be tested was the hybrid Convolutional Neural Network - Bidirectional Long Short-Term Memory, which is denoted as CNN-BiLSTM. It was trained using a Google Colab GPU. The model training was accomplished within 69 epochs and took around 100 minutes to complete.

CNN-BiLSTM performed very well with a test accuracy of 91.3 %, having a macro-averaged F1-score of 0.900 and a weighted F1-score of 0.912. These metrics point to the general effectiveness of the model on the classification task of ten categories described in Table 4.1. Precise per class measurements are given in Table. 4.5.

Table 4.5: CNN-BiLSTM Classification Report

Index	Precision	Recall	F1-score	Support
0	0.94	0.86	0.90	1791
1	0.88	0.78	0.83	1773
2	0.79	0.84	0.80	4028
3	0.79	0.73	0.75	7425
4	0.94	0.94	0.94	3556
5	0.96	0.94	0.95	5584
6	1.00	0.99	0.99	8013
7	0.87	0.94	0.91	4828
8	0.93	0.96	0.95	7603
9	0.98	1.00	0.99	8144
Accuracy			0.91	52745
Macro Avg	0.91	0.90	0.90	52745
Weighted Avg	0.91	0.91	0.91	52745

The model demonstrated strong performance in the overwhelming majority of attack classes, with F1-scores greater than 0.80 for all but a very small number of classes, including excellent results for Normal traffic, Worms, Generic, Fuzzers, Shellcode, Reconnaissance, and Analysis. Performance was comparatively poor only for the Exploits class (F1-score 0.75) and the DoS class (F1-score 0.80), although still indicative of moderate effectiveness. The Backdoor class also had lower performance (F1-score 0.83) than the best performing classes.

Investigation of the modified confusion matrix (Figure 4.4) reveals high accuracy of the model and the issues that still remain. Sometimes, exploits samples were misclassified and were classified as DoS, Normal traffic, or Reconnaissance. DoS samples exhibited a little bit of confusion, particularly in Exploits and Reconnaissance. Sometimes there were attempts to confuse backdoor samples and Exploits. At times, the analysis samples were confused as Exploits. Normal traffic and Worms instances had very high detections with low confusion.

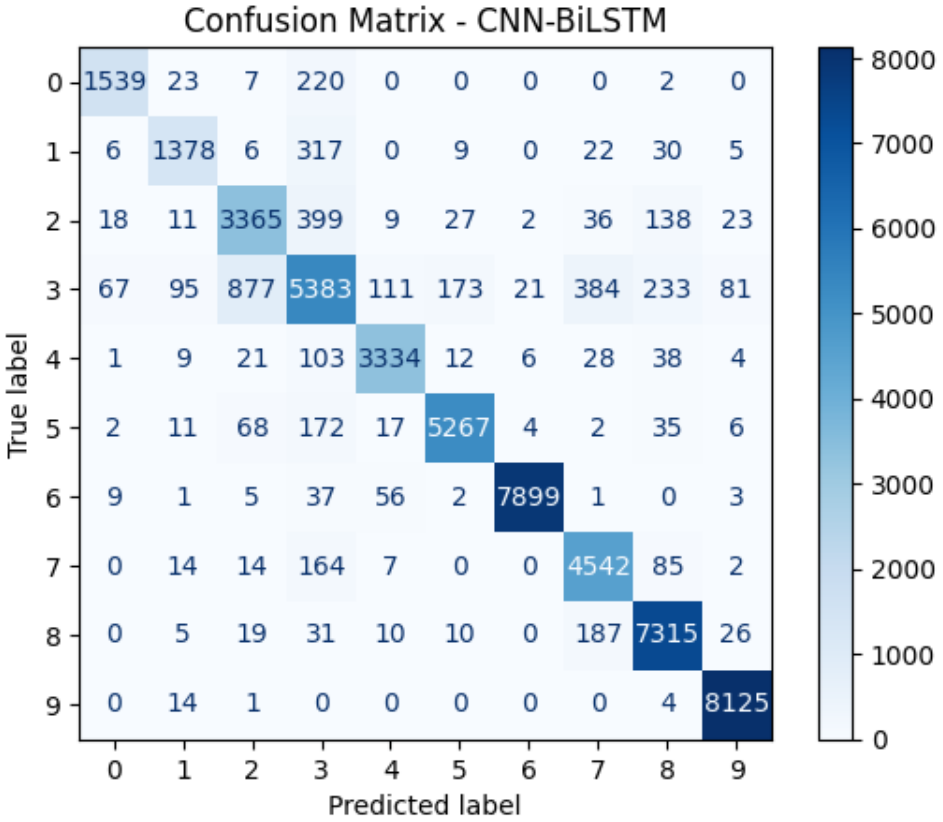


Figure 4.4: Confusion Matrix for the CNN-BiLSTM Model

Finally, when summarizing, it can be concluded that for the purpose of the current research, the CNN-BiLSTM model demonstrated the best general performance among the evaluated architectures, as it managed to detect most attack patterns with a high level of accuracy. Although it showed better performance, there were still small problems, especially when it came to robust distinction between Exploits and DoS attacks. Training process revealed stable learning and effective overfitting mitigation resulting in a strong generalization on the test set.

4.1.5 Binary Classification Results

Besides the multi-class analysis, the performance of the four models was tested on a simpler task of binary classification on what to distinguish between two classes only: Normal traffic and any type of Attack. This gives an idea of the basic capacity of the models to identify deviant traffic from normal traffic apart from a given category of attack. The results of all models show a much better performance than the multi-class task.

The 2D CNN model achieved an accuracy of 98.4% and an F1-score of 0.984 on the binary task (Figure 4.5). It showed excellent separation, with relatively few misclassifications.

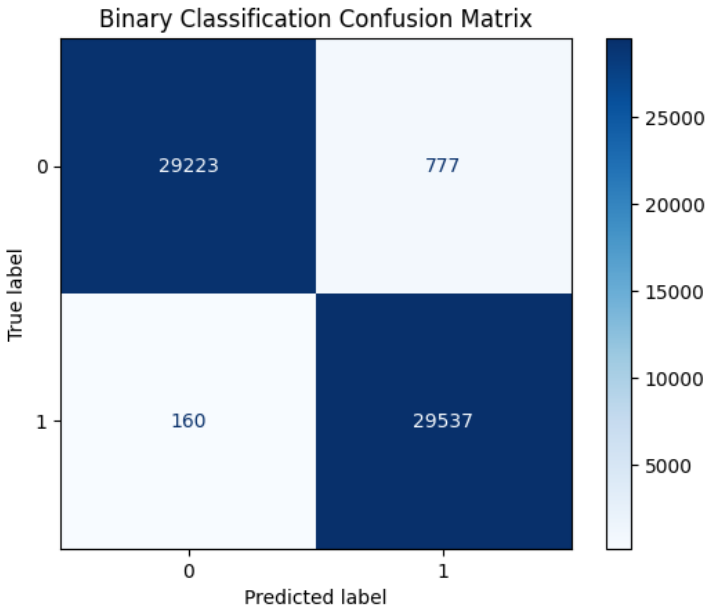


Figure 4.5: Binary Confusion Matrix for the 2D-CNN Model

The MLP model also performed very well, reaching an accuracy of 98.2% and an F1-score of 0.982 (Figure 4.6). While still highly effective, it exhibited slightly more errors compared to the 2D CNN.

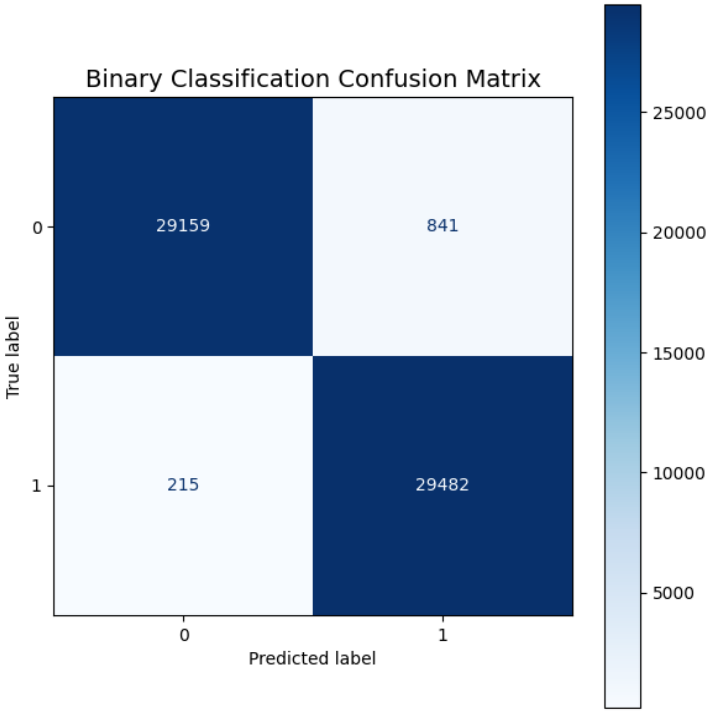


Figure 4.6: Binary Confusion Matrix for the MLP Model

The TabNet model, despite its strong multi-class performance, yielded slightly lower results in the binary setting compared to the other models, achieving an accuracy of 97.2% and an F1-score of 0.972 (Figure 4.7). It recorded the highest number of misclassifications among the four.

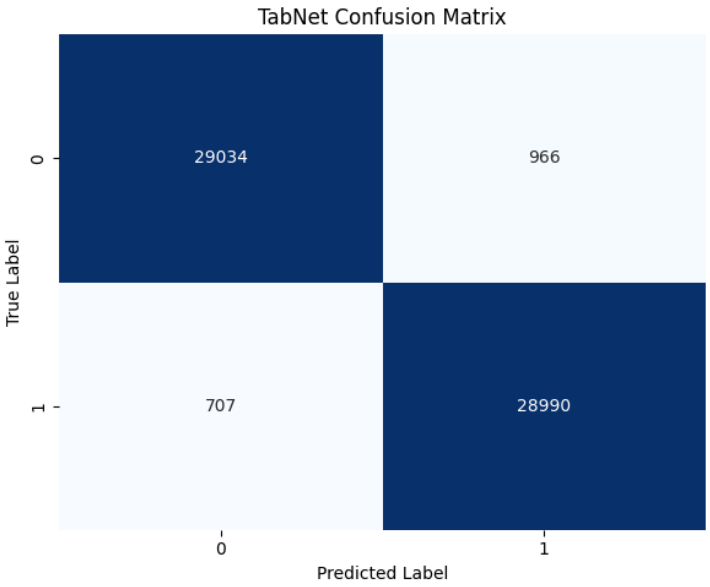


Figure 4.7: Binary Confusion Matrix for the TabNet Model

The CNN-BiLSTM model has shown the best performance in the binary classification task with an accuracy equal to 98.9% and F1-score equal to 0.989 (Figure 4.8). It showed the least misplacements collectively, meaning that its ability in differentiating between normal and general attack traffic was the highest.

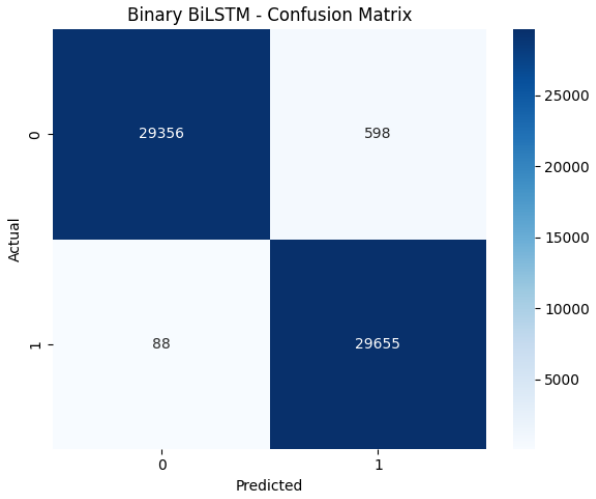


Figure 4.8: Binary Confusion Matrix for the CNN-BiLSTM Model

In short, all four deep learning models performed superbly at the simple binary task of identifying attack traffic versus normal traffic, all with accuracy rates above 97%. CNN-BiLSTM was slightly better than the rest, but TabNet lagged by a small margin, though also at a very high rate. This peak baseline performance indicates that the primary challenge for such models is not anomalous traffic detection in general, but rather fine-grained classification of specific, often related, attack types within the multi-class scenario.

4.2 Comparative Analysis of Models

This section offers a comparative description of the four deep learning models tested—2D CNN, MLP, TabNet, and CNN-BiLSTM—based on their training characteristics, the general performance indicators in the multi-class and binary settings, and the class-specific detection properties employing the class encodings as given in Table 4.1.

An apparent hierarchy took place with regard to the overall multi-class classification performance according to the recent results. While, the hybrid CNN-BiLSTM model showed substantially better result, giving the highest test accuracy (91.3%) and weighted F1-score (0.912), indicating its superior capability in handling the intricate patterns of diverse attack types. And in the second place was TabNet having a robust accuracy of 85.7% and a weighted F1-score of 0.854, showcasing the effectiveness of its attention mechanisms for tabular data. The 2D CNN (81.6% accuracy, 0.808 weighted F1) and MLP (81.5% accuracy, 0.809 weighted F1) models demonstrated significantly better but about the same moderate performance levels as compared to the top two, suggesting a performance plateau for these simpler architectures on this complex dataset. The same ordering applies to the macro-averaged F1-scores, which describes balance in performance over classes: CNN-BiLSTM (0.900) was the first followed by TabNet (0.831) and MLP (0.781) and 2D CNN (0.778) (Table 4.6).

In the simpler binary classification task (Normal vs. Attack), all models performed exceptionally well, achieving accuracies above 97%, which underscores their fundamental ability to distinguish general anomalous behavior. The CNN-BiLSTM again achieved the highest accuracy (98.9%) and F1-score (0.989), reaffirming its robustness across different levels of classification granularity. The 2D CNN (98.4% accuracy/F1) and MLP (98.2% accuracy/F1) also showed excellent results, demonstrating that even less complex models can be highly effective when the task is simplified. Interestingly, TabNet, despite its strong multi-class performance, recorded the lowest binary accuracy (97.2%) and F1-score (0.972) among the four, an unexpected outcome given its sophisticated architecture.

Table 4.6: Comparison of Model Performance and Training Time (Updated)

Model Name	Multi-Class			Binary	
	Acc (%)	Macro F1	Time (min)	Acc (%)	Macro F1
2D CNN	81.6	0.778	≈ 32	98.4	0.984
MLP	81.5	0.781	≈ 34	98.2	0.982
TabNet	85.7	0.831	≈ 24	97.2	0.972
CNN-BiLSTM	91.3	0.900	≈ 100	98.9	0.989

For the case of multi-class detection capability, the same pattern remained steady across the models. All of the models had scored highly for several classes such as Normal traffic, Worms, Generic, Fuzzers, Shellcode, and Reconnaissance attacks, whose scores remained high on every board, suggesting these categories possess more distinct and learnable features. However, the weak points shared by the 2D CNN, MLP, and TabNet models included the difficulty to differentiate among DoS, Backdoor, and Exploits classes, whose F1-scores remained constantly low and mostly confused, indicating potential feature overlap or insufficient model capacity to capture their nuances. The CNN-BiLSTM model largely overcame these challenges with high F1-scores for DoS and Backdoor, with Exploits remaining the weakest category, showcasing its advanced feature learning capabilities.

From the comparison of the models, one can observe different trade-offs that are crucial for practical deployment considerations. TabNet became the fastest model for training (≈ 24 min) offering a considerable increase in accuracy for multi-class classification of the MLP/2D CNN, but to the surprise, it demonstrated the worst performance in the binary setting, which might suggest its optimization for complex feature interactions in multi-class scenarios does not translate directly to simpler tasks. The 2D CNN (≈ 32 min) and MLP (≈ 34 min) had decent multi-class accuracy and excellent binary prediction ability and fairly competitive training times, positioning them as viable options where computational resources are constrained. On the other hand, CNN-BiLSTM gave the highest classification success in both multi-class and binary classifications, requiring many computational minutes (≈ 100 min) as compared to all other methods, highlighting a common compromise between peak performance and resource intensiveness.

Therefore, the more complex hybrid architecture (CNN-BiLSTM) excelled in detection accuracy for both multi-class and binary tasks, though it required more training time, making it suitable for scenarios where accuracy is paramount and resources are available. The TabNet performed well in the multi-class task with the best time efficiency, though it surprisingly underperformed in the simpler binary task, indicating its specialization might not be universally optimal. The 2D CNN and MLP models, while relatively efficient to train, showed lower multi-class accuracy and struggled with challenging attack types like DoS, Backdoor, and Exploits, although their high binary classification performance makes them useful for general threat detection.

Chapter 5

Conclusion and Future Work

This thesis set out on a complex exploration on the use and comparison of the performance of some modern deep learning structures tailored for the challenging field of network intrusion detection. The main goal was to effectively test-run the performance of these advanced computational models in accurately classifying a wide range of attacks on networks as well as regular, normal network traffic. This was made possible because of the large and well-known UNSW-NB15 benchmark dataset that simulates realistic behavior of network and the diverse array of contemporary attack techniques. The procedure involved a data preprocessing pipeline, which was of a rather rigorous and multi-stage character and consecutive application, training, and tuning of four independent deep learning networks, namely the two-dimensional Convolutional Neural Network (2D CNN), the conventional Multi-Layer Perceptron (MLP), the TabNet attention-based and the very sophisticated CNN-BiLSTM. A critical and well-structured analysis of their succeeding performance profiles, in both the multi-class and binary classification class systems, constitutes the gist of this research study. The final chapter is meant to summarize and package essential contributions emerging from this comprehensive experiment and, having these findings and limitations at hand, creates fruitful prospects to inspire developing research for this important field of cybersecurity.

5.1 Main Contributions

This research makes some important contributions to advance knowledge and practical applications of deep learning technology in network intrusion detection systems. The work has systematically taken on the systemic pitfalls of the use of real-world benchmark data in these efforts, specifically the concerns over the quality and imbalance of data, and provided a helpful comparison to the unique capacities and limits of different deep learning paradigms in dealing with this particular challenging task of classification. The robust methodology and in-depth analysis enable a complex reading of model efficacy outside of mere accuracy indices due to implementation implications of real-world IDS deployment. A well characterized enumeration of the output from this project is given as follows:

1. **Comprehensive Data Preprocessing Pipeline:** There was a step-by-step preprocessing approach applied to the UNSW-NB15 dataset, addressing critical problems such as duplicate and missing value management, removal of outlier effects, correction of data skewness, feature engineering, elimination of highly correlated features, and most importantly, removal of extreme class imbalance using a combination of SMOTE and RandomUnderSampler followed by feature selection based on mutual information.

2. **Implementation and Evaluation of Diverse Deep Learning Models:** Four different deep learning architectures namely 2D Convolutional Neural Network, Multi-Layer Perceptron, and hybrid CNN-BiLSTM were successfully implemented, trained and assessed for the multi-class intrusion detection application on processed UNSW-NB15 dataset.

3. **Systematic Comparative Performance Analysis:**An extensive comparative analysis had been performed to compare the performance of the models, in terms of the following metrics; overall accuracy, macro and weighted average F1-scores (taking class imbalance into account), and training time efficiency. This gave quantitative insights as to the relative strengths and trade offs for each architecture.

4. **Identification of Class-Specific Detection Capabilities and Challenges:** The study observed attack categories that have been well detected throughout most of the models (e.g., Generic, Shellcode) and more importantly established a common shortcoming of all the tested architectures in being able to discriminate between the Analysis, Backdoor, and Worms attacks, frequently misclassifying them with each other or Normal traffic.

5. **Analysis of Training Dynamics:** For models similar to the CNN-BiLSTM, the training dynamics were observed which showed how regularization schemes (such as high dropout) affect the stability of the validation and how convergence patterns could mean that effective amelioration of major overfitting is being achieved in spite of training volatility.

5.2 Future Work

Though this extensive research provides useful and numerically justified observations about the comparative performance of some of the modern deep learning models in regards to the Network Intrusion Detection Systems (NIDS), and even on their immense potential, it also voices considerations about some critical routes where further research, fine-tuning and innovational optimizations are not only possible, but are considered necessary to take The current study's long-standing problems, specifically the critical problems faced by most of the tested architectures to perform well across the entire spectrum of fine-grained attack categories and what is even worse, to actually be able to reliably and effectively distinguish between a certain type of attacks with a vague or overlapping characteristics, strongly indicate that a lot of work has to be done. As such, building greater

overall robustness of such detection mechanisms, enhancing their generalizability to disparate and unknown network states, and enhancing the precision of fine-grained attack classification are all issues of significant importance, especially under the ever-changing and complex cyber threat landscape. Thus, straightaway on the well-designed empirical results, the well-specified limits, and the critical appreciation of model performances presented by this current study, the following exact future research directions are set. These guidelines are targeted to tackle the described drawbacks, study new approaches and paradigms of architecture, and eventually support an advanced creation of more effective, smart, and deployable DL-based NIDS solutions for the changing requirements of modern networks.

1. **Cross-Dataset Validation:** Carry forward more experiments to test the trained models or similar structures on other different diversities in NIDS benchmark datasets (for example, CIC-IDS2017, CSE-CIC-IDS2018, CTU-13) to evaluate the generality of the findings and model performance under different network environment and the distribution of attack.

2. **Targeted Strategies for Difficult Classes:** Develop and investigate specialized techniques aimed specifically at improving the detection of the consistently misclassified attack types (Analysis, Backdoor, Worms). This could involve exploring advanced feature engineering tailored to these classes, designing specific modules within hybrid models, or employing ensemble methods focused on resolving ambiguities between these categories.

3. **Exploration of Advanced Deep Learning Architectures:** Explore the use of newer or different types of deep learning architectures that were not covered in this study such as Transformer-based deep learning models (good at capturing long range dependences) or GNNs (graph neural networks) which might be more suitable for capturing the relationships that are intrinsic to the data of network traffic.

4. **Exhaustive Hyperparameter Optimization:** Run even deeper hyperparameter tuning for every architecture based on advanced optimization techniques (e.g., Bayesian optimization, genetic algorithms). Although standard configurations were employed, there is a possibility for further tuning that may open an opportunity for further performance improvement, particularly for the more complex models such as TabNet, as well as CNN-BiLSTM.

5. **Real-Time Performance and Deployment Considerations:** Generalize the evaluation to cover inference time and appropriate computational resources (CPU/GPU usage, memory footprint) to evaluate more pragmatic viability of the deployment of these models into real-time NIDS settings that demand low latency and efficiency.

References

[1] Entesar Alatawi and Ulfat Albalawi. "Harnessing AI for Cyber Defense: Honeypot-Driven Intrusion Detection Systems". In: *Symmetry* 17 (2025), p. 628. DOI: 10.3390/sym17050628. URL: https://doi.org/10.3390/sym17050628.

[2] Chengzhi Zhang et al. "Research on Intrusion Detection Method Based on Transformer and CNN-BiLSTM in Internet of Things". In: *Sensors* 25 (2025), p. 2725. DOI: 10.3390/s25092725. URL: https://doi.org/10.3390/s25092725.

[3] P. Hermosilla et al. "Use of Explainable Artificial Intelligence for Analyzing and Explaining Intrusion Detection Systems". In: *Computers* 14 (2025), p. 160. DOI: 10.3390/computers14050160. URL: https://doi.org/10.3390/computers14050160.

[4] Hesham Kamal and Mahmoud Mashaly. "Advanced Hybrid Transformer-CNN Deep Learning Model for Effective Intrusion Detection Systems with Class Imbalance Mitigation Using Resampling Techniques". In: *Future Internet* 16 (2024), p. 481. DOI: 10.3390/fi16120481. URL: https://doi.org/10.3390/fi16120481.

[5] Md Liakat Ali et al. "Deep Learning vs. Machine Learning for Intrusion Detection in Computer Networks: A Comparative Study". In: *Appl. Sci.* 15.3 (2025), p. 1903.

[6] Shu Feng, Luhan Gao, and Leyi Shi. "CGFL: A Robust Federated Learning Approach for Intrusion Detection Systems Based on Data Generation". In: *Appl. Sci.* 15.5 (2025), p. 2416.

[7] Ghalia Nassreddine, Mohamad Nassereddine, and Obada Al-Khatib. "Ensemble Learning for Network Intrusion Detection Based on Correlation and Embedded Feature Selection Techniques". In: *Computers* 14.3 (2025), p. 82.

[8] Dharani Kanta Roy and Hemanta Kumar Kalita. "Enhanced Deep Autoencoder-Based Reinforcement Learning Model with Improved Flamingo Search Policy Selection for Attack Classification". In: *J. Cybersecur. Priv.* 5.1 (2025), p. 3.

[9] Shehla Gul et al. "WGAN-DL-IDS: An Efficient Framework for Intrusion Detection System Using WGAN, Random Forest, and Deep Learning Approaches". In: *Computers* 14.1 (2025), p. 4.

[10] Y. Zhang, R.C. Muniyandi, and F. Qamar. "A Review of Deep Learning Applications in Intrusion Detection Systems: Overcoming Challenges in Spatiotemporal Feature Extraction and Data Imbalance". In: *Appl. Sci.* 15 (2025), p. 1552.

[11] D.-H. Tran and M. Park. "FN-GNN: A Novel Graph Embedding Approach for Enhancing Graph Neural Networks in Network Intrusion Detection Systems". In: *Appl. Sci.* 14 (2024), p. 6932.

[12] B. Li, J. Li, and M. Jia. "ADFCNN-BiLSTM: A Deep Neural Network Based on Attention and Deformable Convolution for Network Intrusion Detection". In: *Sensors* 25 (2025), p. 1382.

[13] E. Alalwany et al. "Stacking Ensemble Deep Learning for Real-Time Intrusion Detection in IoMT Environments". In: *Sensors* 25 (2025), p. 624.

[14] H. Alavizadeh, H. Alavizadeh, and J. Jang-Jaccard. "Deep Q-Learning Based Reinforcement Learning Approach for Network Intrusion Detection". In: *Computers* 11.3 (2022), p. 41.

[15] C. Strickland et al. "DRL-GAN: A Hybrid Approach for Binary and Multiclass Network Intrusion Detection". In: *Sensors* 24.8 (2024), p. 2746.

[16] A.-G. Mari, D. Zinca, and V. Dobrota. "Development of a Machine-Learning Intrusion Detection System and Testing of Its Performance Using a Generative Adversarial Network". In: *Sensors* 23.13 (2023), p. 6151.

[17] D. Musleh et al. "Intrusion Detection System Using Feature Extraction with Machine Learning Algorithms in IoT". In: *J. Sens. Actuator Netw* 12 (2023), p. 29.

[18] A. Deshmukh and K. Ravulakollu. "An Efficient CNN-Based Intrusion Detection System for IoT: Use Case Towards Cybersecurity". In: *Technologies* 12 (2024), p. 203.

[19] H.-D. Le and M. Park. "Enhancing Multi-Class Attack Detection in Graph Neural Network through Feature Rearrangement". In: *Electronics* 13 (2024), p. 2404.

[20] Fahad Albalwy and Mohammed Almohaimeed. "Advancing Artificial Intelligence of Things Security: Integrating Feature Selection and Deep Learning for Real-Time Intrusion Detection". In: *Systems* 13 (2025), p. 231. DOI: 10.3390/systems13040231. URL: https://doi.org/10.3390/systems13040231.