

**Deep Learning Network Intrusion Detection with the Conv1d-Lstm Model:
Integrating CNN and LSTM For Superior Performance**

CIKAMBASI CIZA LUKOGO

**A Thesis Submitted to The School Of Science And Technology In Partial Fulfillment
For The Requirement Of The Degree Of Master Of Science In Computer Information
System Of Kenya Methodist University**

SEPTEMBER 2024

DECLARATION AND RECOMMENDATION

DECLARATION

This thesis is my original work and has not been presented in any other university.

Cikambasi Ciza Lukogo

Signature Date

RECOMMENDATION

This thesis has been submitted for examination with our approval as university supervisors.

Signature..... Date.....

Dr. Lawrence Mwenda Muriira, PhD

Kenya Methodist University

Signature..... Date.....

Robert Mutua Murungi

Kenya Methodist University

COPYRIGHT

©2024 Cikambasi Ciza Lukogo

All rights reserved. No part of this thesis may be reproduced, stored in any retrieval system or transmitted in any form or by any means, electronically, mechanically, by photocopying or otherwise, without prior written permission of the author or Kenya Methodist University, on that behalf

DEDICATION

I dedicate this research to my family, who have always been there for me. I appreciate your support, love, and encouragement throughout my research.

ACKNOWLEDGMENT

I want to express my gratitude towards all those who helped me successfully carry out my research. I am thankful to God for keeping me healthy, providing me with good strength. Additionally, I wish like to express my gratitude to Dr. Lawrence, and Mr. Robert, for their dedication, guidance, valuable suggestions, and ideas throughout this research proposal. Without their support, this research would not have been possible.

ABSTRACT

The escalating frequency and complexity of cyber-attacks present significant threats to corporate networks, resulting in financial losses, reputational harm, and possible data breaches. Traditional Intrusion Detection Systems (IDS), which rely on predefined signatures and rules, have proven inadequate in addressing these advanced threats due to high rates of both false positives and negatives. This inadequacy necessitates the development of more advanced intrusion detection methods. This thesis introduces a novel AI-based intrusion detection model leveraging deep learning techniques to enhance corporate network security. The proposed model utilizes convolutional neural networks (CNN) and recurrent neural networks (RNN) to analyze network traffic data from the comprehensive CSE-CIC-IDS-2018 dataset, which encompasses a wide array of attack types and provides a realistic representation of modern network traffic. By capturing complex patterns and temporal dependencies in the data, these deep learning algorithms are particularly effective in detecting sophisticated intrusion attempts. A key contribution of this research is the development of a hybrid detection approach that fuses convolutional neural network and recurrent neural network algorithm. This hybrid model enhances detection accuracy and reduces false alarms. The model's performance is rigorously evaluated using metrics such as precision, recall, and F1 score, demonstrating superior detection capabilities with a 99.97% in precision, 99.95% in recall, 99.97% in accuracy, and 99.96% in F1-score outperforming the other models. To address challenges related to data quality, the study incorporates extensive data preprocessing steps, including feature selection, encoding, and scaling. The high computational demands of training deep learning models are mitigated using cloud-based resources. Furthermore, visualization strategies are used to enhance the model's interpretability, offering a glimpse into its decision-making process. The findings of this research have significant implications for network security administrators, researchers, educators, and policymakers. Network security administrators can apply these insights to enhance their defensive strategies against cyber threats. Researchers and educators can leverage the advanced methodologies presented in this study, while policymakers can utilize the findings to inform the development of more effective network security policies and standards. This research advances the field of cybersecurity by proposing and evaluating a novel AI-based intrusion detection model. It underscores the critical importance of integrating advanced AI methodologies into IDS frameworks to protect corporate networks from evolving cyber threats. By improving the accuracy and reliability of intrusion detection systems, this study contributes to the overall security of digital operations in organizations worldwide, highlighting the transformative potential of AI in contemporary cybersecurity.

TABLE OF CONTENTS

DECLARATION.....	II
COPYRIGHT.....	III
DEDICATION.....	IV
ACKNOWLEDGMENT.....	V
ABSTRACT.....	VI
LIST OF TABLES.....	X
LIST OF FIGURES.....	II
LIST OF ABBREVIATIONS AND ACRONYMS.....	III
CHAPTER ONE.....	1
INTRODUCTION.....	1
1.1 Background of the study.....	1
1.2 Statement of the Problem.....	5
1.3 Objectives.....	6
1.4 Justification of the Study.....	7
1.5 Limitations of the Study.....	8
1.6 Delimitation of the Study.....	8
1.7 Significance of the Study.....	9
1.8 Assumptions of the Study.....	10
1.9 Introduction Summary.....	10

CHAPTER TWO	11
LITERATURE REVIEW	11
2.1 Introduction.....	11
2.2 Theoretical framework Intrusion Detection Systems	11
2.3 Types of IDS.....	15
2.4 The current state of the art	18
2.5 Use of Dataset in Machine Learning and Deep Learning.....	27
2.6 The Gap in the Literature.....	29
2.7 The proposed model Architecture.....	30
2.8 Literature Review Summary	32
CHAPTER THREE.....	33
METHODOLOGY	33
3.1 Dataset.....	33
3.2 Tools and Environment.....	33
3.3 Model development	35
3.4 Model evaluation	42
CHAPTER FOUR.....	44
DISCUSSION AND RESULT	44
4.1 Exploratory Data Analysis (EDA).....	44
4.2 Data Preprocessing.....	46
4.3 Model Architecture	52

4.4 Results.....	62
CHAPTER FIVE	78
SUMMARY, CONCLUSIONS AND RECOMMENDATIONS	78
5.1 Introduction.....	78
5.2 Summary of the study findings	78
5.3 Conclusion	79
5.4 Recommendations.....	79
5.5 Future Work.....	81
REFERENCES.....	83
APPENDICES	92

LIST OF TABLES

Table 2.1 Commonly Used Ids Datasets and Types of Attacks Featured	28
Table 3.1 One-hot encoder	39
Table 3.2 label encoding	40
Table 4.1 CSE-CIC-IDS2018: Number of instances for each type of attack.....	45
Table 4.2 The set of characteristics used for the model	48
Table 4.3 Data split	52
Table 4.4 Confusion Matrix For Each Attack Type.....	73
TABLE 4.5 Evaluation Metrics	75

LIST OF FIGURES

Figure 2.1 Host-based IDS	16
Figure 2.2 Network-based IDS	17
Figure 2.3 Machine Learning Techniques	20
Figure 2.4 The proposed model Architecture	31
Figure 3.1 Architecture for model development	36
Figure 3.2 Data Preprocessing	38
Figure 3.3 split datasets.....	41
Figure 4.1 CSE-CIC-IDS2018: Number of instances for each type of attack	46
Figure 4.2 The architecture of Convolution Neural Network Model	55
Figure 4.3 Architecture of a Recurrent Neural Network	58
Figure 4.4 Architecture of Hybrid network	61
Figure 4.5 Models Training and Validation Performance	65
Figure 4.6 Convolution Neural Network Confusion Matrix.....	69
Figure 4.7 Recurrent Neural Network Performance confusion matrix	70
Figure 4.8 Hybrid Model Performance confusion matrix.....	72
Figure 4.9 Model's Performance	76

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
ANN	Artificial Neural Network
ANN	Artificial Neural Network
CNN	Convolutional neural network
CSE-CIC-IDS-2018	A collaborative project between the Communications Security Establishment (C.S.E.) & the Canadian Institute for Cybersecurity (C.I.C.)
DDoS	Distributed Denial of Service (DDoS)
DL	Deep learning
FCM	Fuzzy Cognitive Map
HIDS	Host-based Intrusion Detection System
IDS	Intrusion detection systems
IoT	Internet of Things
IPS	Intrusion Detection System
IT	Information Technology
k-NN	k-Nearest Neighbors
LSTM	Long Short-Term Memory
ML	Machin Learning
NIDS	Network-based Intrusion Detection System
NSL-KDD and ISCXIDS2012	Intrusion detection Datasets
PCA:	Principal Component Analysis
RNN	Recurrent Neural Network
SSH	which stands for Secure Shell
The KDD Cup 1999	Knowledge Discovery and Data Mining Cup 1999

+

CHAPTER ONE

INTRODUCTION

1.1 Background of the study

In today's digital age, network security has become very important for any organization that depends on information technology to conduct its operations. According to Gřivna and Drápal (2019). Malicious cyberattacks have the potential to disclosure the secrecy of information, breaching the integrity, and accessibility of data and systems, resulting in significant financial losses, harm to reputation, and legal responsibilities (Gřivna & Drápal, 2019).

In March 2018, the personal data of 50 million Facebook users were exposed to the internet (Rehman, 2019). “In 2020 the average expense associated with a data breach was \$3.86 million, and it took an average of 280 days to detect and contain a breach” (*What's New in the 2019 Cost of a Data Breach Report*, 2020). Detecting and preventing unauthorized or malicious activities that disrupt the usual operations of a network or system, known as intrusions, is a primary challenge in network security.

Intrusions can be initiated by internal or external actors, and these actors can employ a variety of techniques to damage a computer system or an organization network. These techniques include denial-of-service attacks, phishing, brute-force attacks, exploitation of system vulnerabilities, or deployment of malware (Contributor, 2021, Contributor, 2022), according to a report by Beijing News on November 16, 2020. China's YTO Express admitted that five employees had sold their internal accounts to criminal organizations, which resulted in an unauthorized disclosure of around 400,000 pieces of personal information belonging to users (Esg, 2023). The leak of

sensitive data has become the most common security incident in the world, and it is essential to find new methods to address this issue.

An IDS is a computer program or hardware that monitors network traffic or system logs to identify suspicious activities and provide alerts accordingly (Fortinet, n.d.). IDS is crucial in enhancing cybersecurity by identifying potential threats and helping protect systems and networks.

However, traditional Intrusion Detection and Prevention Systems face several limitations when it comes to sophisticated cyberattacks. Most intrusion detection and prevention systems depend on signatures, which are pre-defined rules or patterns that match known attacks

Signature-based detection is effective against well-known threats, but it cannot identify modified attacks that evade the rules. Moreover, signature-based detection often generates a high number of false positives, which occur when IDS alerts on activity that is not a threat, and false negatives when the systems fail to detect a malicious action that has occurred (Khraisat et al., 2019).

Traditional IDS are essential for the monitoring of network traffic and system events to identify and alert suspicious activities. However, these systems face significant challenges when it comes to sophisticated cyberattacks. Most intrusion detection and prevention systems depend on signature-based detection, which are predefined rules or patterns to recognize known threats, IPS are adept at handling familiar attacks but are impeded by their inability to detect modified threats that do not match existing signatures. This limitation is compounded by the prevalence of false positives erroneous alerts on benign activities and false negatives missed detections of actual

malicious actions. The research by Khraisat et al. (2019) highlights the frequency of such inaccuracies, which can undermine the trust in and efficacy of these security measures.

Moreover, traditional IDPS are often ill-equipped to handle zero-day vulnerabilities, which are newly discovered security flaws that have not yet been catalogued in signature databases. This leaves a critical gap in defense until signatures are updated to include these new threats. To mitigate these shortcomings, some IDPS have adopted anomaly-based detection strategies that learn and establish a baseline of normal behaviour, thereby enabling the identification of deviations that may signify an attack. However, this method is not without its shortcomings, as it can lead to false positives.

One effective solution involves a hybrid technique integrating signature-based and anomaly-based detection methods. This hybrid system aims to improve threat detection accuracy while minimizing false alarms by utilizing the strengths of each detection method. As cyber threats continue to evolve, it is imperative for cybersecurity professionals to adapt their strategies, accordingly, ensuring a balanced approach to detection that maximizes security while minimizing disruptions caused by false alerts

False positives can reduce the efficiency and reliability of IDS. Over time, cyber-attacks have become increasingly advanced. This has resulted in an increased frequency of attacks, underlining the importance of implementing more robust cybersecurity measures to prevent such threats (Park et al., 2022). With the increasing occurrence of advanced cyber-attacks, there is a demand for more advanced intrusion detection systems, such as AI-based IDS and IPS, to provide better protection against

these threats. Recent research has been dedicated to examining the incorporation of artificial intelligence (AI) technology into IDS, which has resulted in the creation of AI-based intrusion detection systems that have shown outstanding efficacy in detecting and preventing potential network intrusions (Park et al., 2022).

Artificial Intelligence (AI) techniques have been suggested to solve the shortcomings of IDS and IPS. These AI approaches could learn from data and adjust to changing conditions without being limited by pre-defined rules or human involvement. In addition, AI could enhance the accuracy of IDS and IPS by decreasing the rate of false positives as well as false negatives rate, which are security alerts that indicate malicious activity that is not detected by the system. Furthermore, AI could offer insights into the origin and nature of intrusions, as well as recommendations for response and recovery (Jayalaxmi et al., 2022).

Machine Learning and Deep Learning approaches have become popular in detecting and preventing intrusions. These two subfields of Artificial intelligence focus on creating systems that are able to learn from massive dFata and perform tasks like Natural language processing, categorization, regression, clustering, and anomaly detection. By leveraging a large volume of data from numerous sources, such as network packets, system logs, user behaviours, and external threat intelligence, ML and DL algorithms could be used to train models that can identify intrusions based on extracted features or patterns from the data. In addition, they can manage complex and non-linear relationships within the data, as well as high-dimensional and heterogeneous data types (Janiesch et al., 2021)

According to Janiesch et al. (2021), while Deep learning and machine learning are promising intrusion detection approaches, they present several challenges and unresolved issues. For instance, they require adequate and representative data to train the model, which may only sometimes be available due to privacy or ethical concerns. Additionally, they must contend with data quality issues, such as noise, outliers, imbalance, or redundancy, which can impact their overall performance or generalization capabilities. Furthermore, ensuring the interpretability of the model can take time due to its complexity. Lastly, ML and DL models must also provide their security against Advanced attacks, which involve purposeful attempts to trick the models by employing prepared inputs. Therefore, there is a demand to develop an AI-based intrusion detection model that can address these challenges while enhancing corporate network security. This research introduces an innovative approach to improving the safety of corporate networks using an AI-based intrusion detection model. This model leverages cutting-edge artificial intelligence techniques and is specifically designed for network intrusion detection. By utilizing this model, corporate networks could be safeguarded against potential threats more effectively than with traditional methods.

1.2 Statement of the Problem

As Organizations rely primarily on corporate networks for their operations, the frequency and increasing complexity of cyber-attacks pose a significant threat to this corporate network, resulting in financial losses, reputational harm, and data breaches.

Traditional IDS are no longer adequate in detecting and preventing advanced cyber-attacks. Furthermore, they experience false positives, which trigger unnecessary alarms, and false negatives, which compromise the system's security, and undermine

the performance and accuracy of IDS. As a result, the effectiveness and precision of the system are significantly affected, making it difficult to prevent cyber threats. (Liang et al., 2019).

In an era where corporate networks are the backbone of organizational operations, the escalation in both the sophistication and frequency of cyber-attacks represents a significant threat to this corporate network. These incursions jeopardize the integrity of corporate networks, causing serious repercussions like financial losses, reputational harm, and data breaches. Traditional intrusion detection systems (IDS), which form the first line of defense, are proving to be increasingly inadequate in detecting and preventing advanced cyber-attacks. They present limitations, including false positives that result in unwarranted alarms, and false negatives that allow security breaches to go undetected (Raponi et al., 2019). These shortcomings not only jeopardize the security of the system but also impede the performance and accuracy of IDS. Consequently, the overall effectiveness and precision of these systems are significantly diminished, presenting a substantial challenge in the effective prevention of cyber threats. This predicament necessitates the development of a more sophisticated and adaptive approach to intrusion detection. This research introduced an AI-based intrusion detection model to address these challenges by leveraging the capabilities of artificial intelligence and deep learning to increase the efficacy of intrusion detection mechanisms within secured corporate networks.

1.3 Objectives

The study had the following specific objectives:

1. To apply an AI-based model in corporate networks for intrusion detection: The proposed model included an artificial intelligence method with an intrusion detection capability to identify potential security risks.
2. To assess the performance of the proposed model: The model was tested and assessed to see its effectiveness in detecting cyber threats.
3. To Examine the performance of both algorithms, CNN and RNN which were tested on the CSE-CIC-IDS-2018 dataset.

1.4 Justification of the Study

This research was vital because cybersecurity is a critical challenge in today's world. Cyberattacks can cause significant damage to corporate networks and compromise the confidentiality of organization information. Therefore, it is essential to apply an effective solution to safeguard corporate networks from cyberattacks.

This research was essential to various stakeholders, such as network security Administrators, researchers, Network security educators, and Network security policymakers. Network security administrators could benefit from this study by gaining insights and recommendations for improving their network security using AI techniques. Network security researchers could benefit from this study by advancing their knowledge and understanding of AI techniques for network security. Network security educators could benefit from this study by incorporating the findings and implications of this study into their curriculum and pedagogy. Network security policymakers could use this study to develop network security policies and standards.

This research was significant as it provided a new and improved method of network security that could detect security threats. Furthermore, this study contributed to the

field of network security by providing a novel and comprehensive AI-based intrusion detection model that addressed the limitations of traditional network security solutions.

1.5 Limitations of the Study

This study faced several challenges that could have affected the research's outcomes. One of the challenges was the availability of datasets for training and testing the model. This research used the available public datasets, such as CSE-CIC-IDS-2018. Additionally, AI models for intrusion detection can be complex and require a substantial number of computational resources to train and test. Therefore, to address this challenge, the study utilized cloud-based computing resources such as Google Collaboratory to support the model evaluation. Moreover, ethical, and legal issues may have been associated with using AI in network security.

The research ensured compliance with ethical guidelines and regulations related to data privacy and AI development to align with ethical and legal requirements.

Another challenge was the interpretability of AI models, which made it difficult to better understand the model's process of decision-making. The study explored visualization techniques and built models that provided detailed explanations of their decision-making process to increase the model's interpretability.

1.6 Delimitation of the Study

This research might have some delimitations that enhanced its feasibility and validity.

One of the delimitations of this research was that the study focused on applying an AI-based intrusion detection model for corporate networks. The research did not cover other aspects of network security, such as physical security, encryption, authentication, or firewall technology.

Additionally, the research utilized Deep learning techniques like convolutional neural networks as well as recurrent neural networks. It did not employ AI techniques that are not applicable or appropriate for intrusion detection tasks, such as computer vision and speech recognition.

Moreover, the research employed realistic network traffic datasets that contain various types of intrusions to train and test the model. The research selected publicly available datasets and avoided using outdated or irrelevant datasets for this research.

1.7 Significance of the Study

The significance of this study rests in its potential to address one of the most pressing challenges of the digital age: cybersecurity. As organizations depend more and more on computer networks for their daily operations, ensuring the integrity, confidentiality, and availability of information has become a daunting task. By using deep learning algorithms to train a model that could identify patterns and intrusions in network traffic, this research could enable real-time detection of potential threats. This, in turn, could allow security personnel to respond swiftly, thereby decreasing the risk of successful attacks and improving the overall security of the corporate network.

Furthermore, this research could contribute to the development of best practices and guidelines for implementing and deploying AI-based models for intrusion detection. These guidelines could be invaluable to organizations across various industries and sectors, helping them to safeguard their networks more effectively. In essence, this study could advance our knowledge in the field of cybersecurity by proposing and evaluating an AI-based intrusion model. By doing so, it could play a crucial role in

enhancing network security and, by extension, the security of digital operations in organizations worldwide.

1.8 Assumptions of the Study

This study made some assumptions as follows:

1. The AI-based intrusion detection model was valid and reliable for detecting intrusions within corporate networks.
2. The realistic network traffic datasets, which were used to train and test the AI-based intrusion detection model, represented the actual network traffic encountered in various organizations.

1.9 Introduction Summary

In today's digital landscape, organizations face increasing threats from sophisticated cyberattacks that traditional Intrusion Detection Systems (IDS) struggle to detect, often resulting in false positives and negatives. The primary objective of this study is to develop a more advanced and adaptive IDS using artificial intelligence (AI) and deep learning techniques. The proposed solution involves integrating AI-based models, particularly Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), to enhance the accuracy and efficiency of detecting and preventing network intrusions. This approach aims to address the limitations of traditional IDS, providing a more robust defense against evolving cyber threats.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

The aim of this chapter is to investigate how Artificial Intelligence (AI) models can be used to enhance network security by detecting intrusions on company networks. The review assesses various AI techniques utilized in intrusion detection models.

2.2 Theoretical framework Intrusion Detection Systems

IDS have been widely adopted as a protective measure in cybersecurity. These systems offer a more advanced layer of security Compared to standard firewall technology. While firewalls serve as a basic barrier against unauthorized access, intrusion detection and intrusion prevention systems (IPS) provide a deep level of protection. They actively monitor network traffic for suspicious activity and potential threats, thereby significantly enhancing the security posture. By employing sophisticated algorithms and patterns, IDS and IPS can detect and prevent a broad variety of cyber threats, making them an essential component in safeguarding corporate networks.

Traditional IDS uses signature-based techniques to recognize known attacks, but these methods can be circumvented by skilled attackers (Fortinet, n.d.).

Intrusion detection systems (IDS) were initially developed in the 1980s to identify malicious activities in computer networks (Anderson, 1980). Initially, IDS was based on some rules that used pre-defined signatures to detect known attacks. However, these systems had limitations in their ability to identify new attacks. In the 1990s, anomaly-based detection systems were developed, which used statistical models to detect

deviations from normal network behavior. These systems were more effective in detecting new attacks but suffered from high false-positive rates.

Munivara et al. (2017) introduced a novel approach to identify DDS attacks at the application layer. The authors created a new set of tools using the absolute time interval standard to build and test their model. Their approach involves three algorithms Cuckoo, Shark, and Bat then they employ the Cuckoo Binary Clustering strategy to decrease the costs associated with these algorithms and improve prediction accuracy.

Al-Yaseen et al. (2016) created a model capable of efficiently handling intrusion detection challenges in data. Their objective was to classify network data as normal or abnormal accurately. To ameliorate the detection of both well-known and new attacks, a multi-level hybrid model was suggested. This model incorporates support vector machine technology to enhance its efficiency. Additionally, the researchers recommended K-means algorithm for creating a quality dataset to train the model that resulting in better classifier performance. The K-means algorithm enabled the creation of smaller training datasets that represent the whole original training dataset, decreasing the time for training the model and boosting the performance of the IDS. To assess their model, the researchers used the widely used KDD Cup 1999 dataset and compared their results with other techniques using the same dataset. Their proposed model showed a high level of effectiveness in detecting attacks, achieving the best performance accuracy of 95.75%.

Mazini et al. (2019) proposed a dependable technique for network-based anomaly detection and assess its performance using NSL-KDD and ISCXIDS2012 datasets.

The article highlights how the large and unbalanced nature of datasets can affect the performance of the model, as conventional data-mining algorithms tend to overlook the minority class, resulting in inaccurate classification. The proposed approach utilizes the AdaBoost meta-algorithm to handle unbalanced data and the Artificial Bee Colony (ABC) to optimize IDS problems. The algorithm also selects the most relevant features to identify network connections. The outcomes demonstrate that the proposed technique exhibits high precision in classifying various attack classes.

An IDS is a sophisticated program application developed for monitoring network traffic and identify suspicious or malicious behaviour. It operates by analyzing data packets that traverse the network, looking for patterns or anomalies that may indicate a security threat. When the system identifies any security risks, it promptly sends an alert to the IT personnel or to the security teams, who can then take the necessary actions to investigate and mitigate the threat. The functionality of an IDS is not limited to a single point of defense; it can detect suspicious activity at both the network level and the host level.

Additionally, some IDS can prevent intrusions by blocking malicious or suspicious traffic, which is referred to as an IPS. An IPS not only detects threats but also has the ability to prevent intrusions by blocking malicious or suspicious traffic before it can cause harm (Fortinet, n.d.).

For instance, if an IPS identifies a data packet that corresponds to a recognized malware's signature, it can automatically reject the packet and prevent it from reaching its intended destination. Given the rapid evolution and complexity of cyber threats in today's digital environment, this level of automation and reactivity is essential.

The distinction between IDS and IPS is significant: while an IDS serves as a detection and alert system, an IPS acts as a control system that actively intervenes to protect the network (Ashoor & Gore, 2011).

Usually Intrusion detection systems use two main techniques: anomaly-based and signature-based detection

2.2.1 Signature-Based IDS.

According to Janis (2021), a Signature-based IDS is a critical security mechanism that operates by detecting predefined patterns, known as “signatures,” which are indicative of malicious activity in network traffic or system files. The IDS employs a database of established attack signatures to scan incoming network traffic or system files for any matches. If a match is found, the IDS activate an alert or prohibit the traffic to avert the attack from happening.

Jyothsna et al. (2011) state that Signature-based IDs are proficient in identifying already known attacks, particularly those that exploit well-known vulnerabilities in programs or operating systems. However, they may need to be more efficient when it comes to new attacks, which do not have a pre-existing signature. Moreover, attackers can avoid signature-based IDs by making changes to their attacks to avoid detection. The authors noted that signature-based IDs have several limitations, including the inability to detect attacks with no pre-existing signatures in the databases. Moreover, for every attack, It is necessary to create a unique signature, because signature-based IDS cannot identify new attacks (Jyothsna et al., 2011).

2.2.2 Anomaly-Based IDS

Anomaly-based detection systems are a critical component in safeguarding network security. They operate by establishing a baseline of normal network behaviours, which is derived from a set of pre-defined parameters. These parameters are meticulously crafted by network administrators who utilize their expertise to determine what constitutes normal operations within the network's environment. The process of defining these behaviors can be achieved through direct configuration or through machine learning algorithms that adapt over time to the network's changing patterns (Jyothsna et al., 2011).

According to Janis (2021), anomaly-based detection is intended to identify unfamiliar attacks, such as new malware, and adjust to them using machine learning. Anomaly-based IDS has advantages in detecting previously new threats and identifying issues caused by non-malicious factors, such as misconfiguration and program bugs. However, while anomaly-based detection systems are powerful, they are not without their challenges. One major problem is the tendency to produce false positives which occurs when the system mistakenly identifies regular behavior as malicious, or a false negative that happens when an actual threat goes undetected (Janis, 2021). These inaccuracies can be attributed to the dynamic nature of network behaviors and the complexity of distinguishing between benign anomalies and genuine threats.

2.3 Types of IDS.

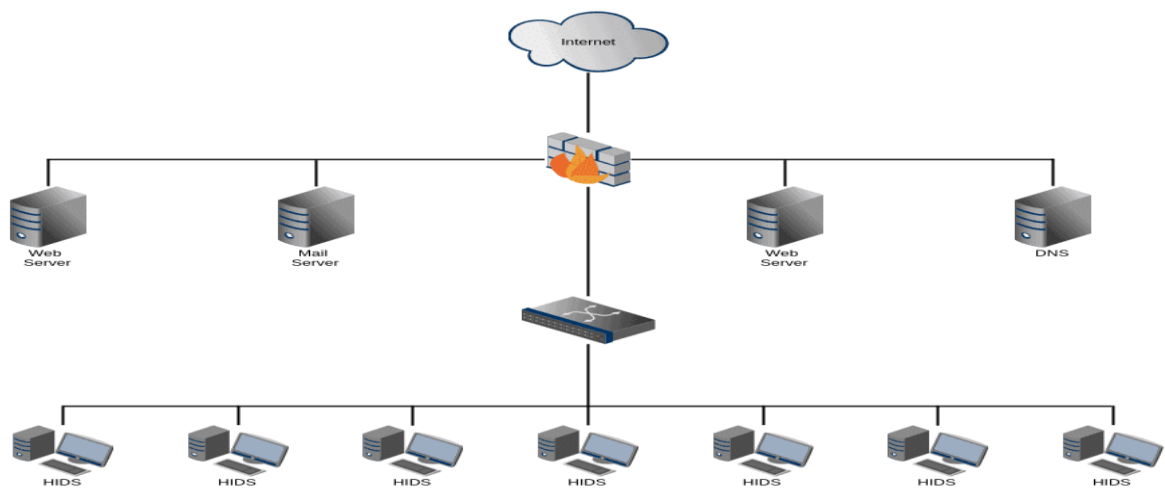
Intrusion detection systems are important in network security because it helps safeguard computer systems and networks from unauthorized access. There are different types of IDS.

2.3.1 Host-based IDS

A Host Intrusion Detection System (HIDS) is security systems that are installed on a computer (host) in a network to detect intrusions. HIDS monitors the activities of the host and sends an alert to IT personnel or security teams if it detects any suspicious or abnormal behaviour (vanin et al., 2022).

Figure 2.1

Host-based IDS



Note. Host based intrusion detection system architecture From Wikimedia.
(https://commons.wikimedia.org/wiki/File:Host_based_intrusion_detection_system.png). Copyright 2016 by Rhouere.

Many researchers have begun to investigate the use of machine learning to detect intrusions on hosts. Host-based intrusion detection systems analyze a host's characteristics and actions, including monitoring data like system logs and network traffic information (Gupta and Mamtora 2012; Deshpande et al., 2018), to detect potential threats. By using machine learning techniques, HIDS can efficiently investigate large amounts of data and pinpoint anomalies or suspicious behaviors that may indicate a security breach. This has the potential to significantly increase the

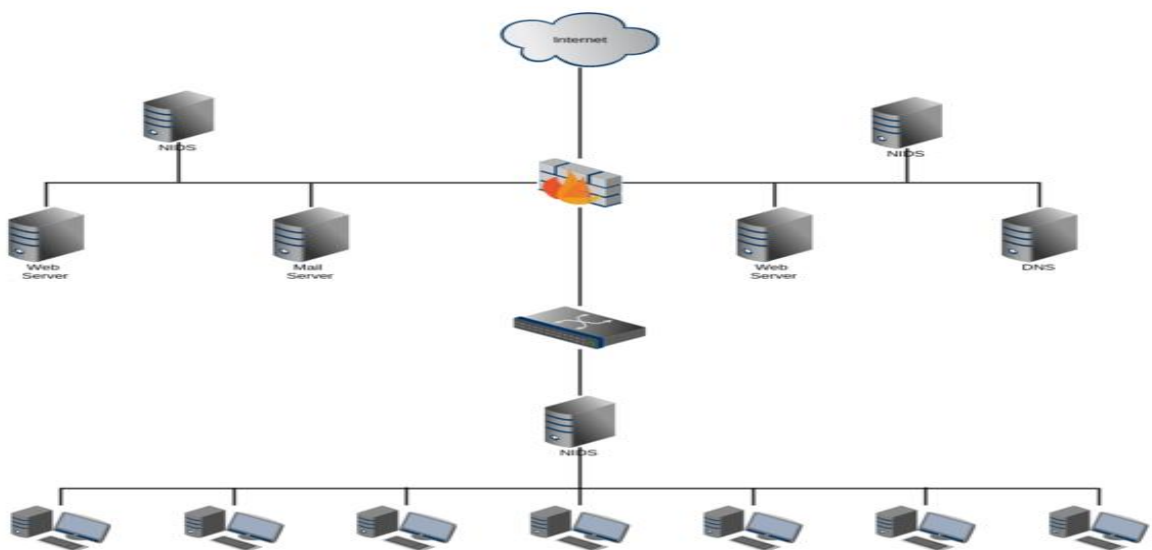
security of computer systems and networks, particularly in today's constantly changing threat environment.

2.3.2 Network-based IDS

A Network-based IDS is an intrusion detection system positioned within a network to monitor the overall network traffic and detect any potentially malicious activities. NIDS provides a comprehensive view of the network, allowing it to detect attacks from any host within the network. NIDS is a crucial component of network security for most organizations (Vanin et al., 2022).

Figure 2.2

Network-based IDS



Note. Network based intrusion detection system architecture From Wikimedia.
(https://commons.wikimedia.org/wiki/File:Network_based_intrusion_detection_system.png). Copyright 2016 by Rhouere.

Numerous studies have explored the potential of machine learning techniques for NIDs. For example, Qassim et al. (2016) suggested an anomaly-based IDS that could recognize network traffic that deviates from the norm and generate alarms accordingly.

However, accurately differentiating between false positives and true alarms remains a significant problem. The authors proposed a two-step approach. They recommended a set of network traffic characteristics that were most useful in identifying network anomalies. Next, the authors proposed a system that uses machine learning algorithms, and an anomaly detection system based on packet headers in order to classify activities and detect malicious behavior. They found that the system was effective but suggested that incorporating additional machine-learning approaches could increase accuracy even further.

2.4 The current state of the art

The advancement of artificial intelligence (AI) techniques has resulted in the emergence of AI-based intrusion detection models with the potential to detect new and advanced attacks (Jayalaxmi et al., 2022).

Over the last few decades, there has been a rise interest in employing artificial intelligence (AI) to improve network security. Researchers have proposed various models that utilize machine learning approaches to detect and prevent cyber-attacks.

Early studies employed ML Algorithms, including linear regression, k-Nearest Neighbors (k-NN), decision tree, support vector machine (SVM) and Naïve Bayes to develop intrusion detection and prevention systems (Liang et al. 2019).

Liang et al. (2019) introduced a hybrid approach that combines clustering with SVM to detect attacks. Their method involves using K-means clustering to divide the data into different sets and then applying SVM to each. They used NSL-KDD datasets to test their approach and achieved 99.450% accuracy. Additionally, the solution was computationally efficient and required less processing time than SVM algorithms that

use various parameters. However, the authors did not report on the accuracy of individual attack classifications, which could be a subject for further research.

Kanimozhi and Victoire (2021) published a study on detecting cloud intrusions using a unique method. The technique involved using an oppositional fuzzy C-mean algorithm and involved several steps, such as data preprocessing, feature selection, normalization, using logistic regression, and applying FCM clustering models. The team used the fuzzy C-means technique to split the dataset into C clusters. Following clustering, the researchers performed cluster expansion to eliminate redundant clusters. The proposed model was tested on multiple datasets, including CICIDS2017, and achieved an 80% accuracy rate.

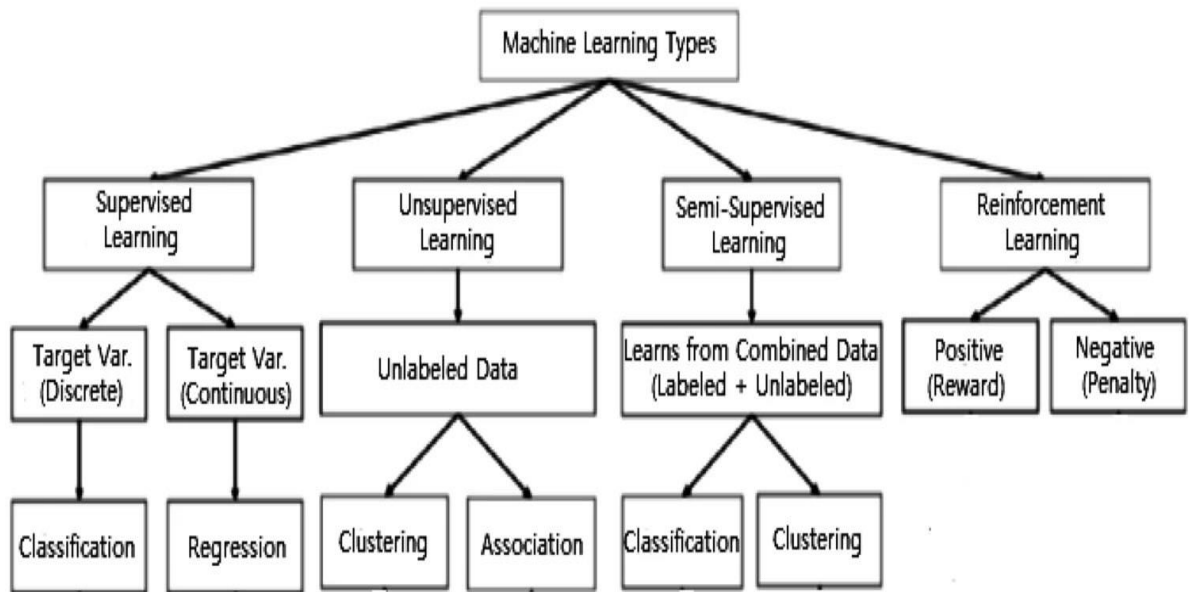
However, more recent research has demonstrated that deep learning algorithms, yield even better results in intrusion detection (Kimanzi et al., 2024). Xiao et al. (2019) developed a technique for detecting network intrusions using a convolutional neural network (CNN). They utilized two techniques, Auto-Encoder, and a principal Component Analysis, for feature extraction. These methods reduced the data dimensions by eliminating insignificant information through neural network hidden layers. The data was then transformed in a dimensional matrix and input into the model for training, which was optimized using backpropagation algorithms. To test the model's effectiveness, the researchers used the KDDcup99 dataset and achieved an accuracy of 94%. In contrast to DNN and RNN models, the CNN model performed slightly better. However, it had a lower observation rate.

2.4.1 Machine Learning Techniques

According to Mohammed et al. (2016), four techniques are used in machine learning, as presented in Figure 3.

Figure 2.3

Machine Learning Techniques



Note. This figure was produced by Sarker in 2021, and it summarized the different type of machine learning techniques. From “Machine Learning Model Development for Screening Potential Entrepreneurs in the B40 (Bottom 40%) for Targeting Assistance” by Gopal, Sagarán & Nair, Sulochana, 2016, *International Journal of Academic Research in Business and Social Sciences*. 11. 10.6007

/IJARBSS/v11-i12/11896. 2021. Copyright by CC.BY-NC-ND 2.0

Supervise Machine Learning

Supervised learning is a technique where a model is trained using data that has been labeled with known input and output pairs. By analyzing patterns in the labeled data, the model learns to map inputs to outputs. Some common examples of supervised learning include regression, classification, and recommendation systems, all of which use labelled data to predict future outputs (Han et al., 2011).

Rosa et al. (2021) also suggest that supervised machine learning (ML) models are typically tailored for a specific scenario, whether for a particular process or a single communication protocol. There is a broad range of supervised ML algorithms, but this thesis focuses on the most commonly used ones in the classification category. These include Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Artificial Neural Networks (ANN), Naïve Bayes, Decision Tree (DT), and Random Forest (RF). While supervised learning is effective for detecting known attacks, it is less suitable for identifying new ones.

Unsupervised Machine Learning

This is a technique where a model is trained on input data that lacks any corresponding output information (Sarker et al., 2020). Unsupervised learning aims to show the data distribution and the correlations between variables without making a distinction between the variables that are observed and the variables that need to be predicted. (Han et al., 2011).

Semi-supervised Learning

Semi-supervised learning is an approach that uses both labelled and unlabeled data to combine supervised and unsupervised learning approaches (Han et al., 2011; Sarker et al., 2020). It is somewhere between supervision along unsupervised supervision. Semi-supervised learning can be especially helpful in situations where labelled data is scarce but there is an abundance of unlabeled data (Mohammed et al., 2016).

Reinforcement Machine Learning

With this type, a machine learning model is trained to make decisions based on its environment's input. The model receives feedback, which it uses to adjust its behaviour and improve its decision-making. The ultimate objective of reinforcement learning is

to maximize the cumulative reward over a prolonged period by learning from experience through trial and error (Mohammed et al., 2016).

Reinforcement learning is when the mode learns from its interactions with the environment and makes optimal decisions in complex situations. For instance, a reinforcement learning model can be trained to play a game by learning from its past experiences and taking actions that lead to maximum rewards, or it can be used to control the movements of a robot by receiving feedback from its sensors and adjusting its actions accordingly.

Machine learning has been utilized by researchers to identify intrusions on the host or network side.

2.4.2 Deep Learning Approaches for IDS.

In recent years, the field of intrusion detection has seen significant advancement through the adoption of deep-learning techniques. Signature-based detection systems operate by matching observed activities against a database of known intrusion signatures. These systems excel at detecting known threats but are less effective against novel or evolving attacks. On the other hand, anomaly-based detection systems are designed to establish a baseline of normal network behaviour, which allows them to identify any unusual patterns that could potentially indicate a security breach. Deep learning methods can be applied to both types of detection due to their ability to extract higher levels of nonlinear relationships within the data, enabling the identification of benign activity deviations.

In the field of cybersecurity, a wide range of deep learning algorithms have been employed to enhance intrusion detection systems. Each architecture brings unique capabilities to address different aspects of intrusion detection.

Recurrent Neural Networks (RNNs): RNNs, including their variant Long Short-Term Memory (LSTM) networks, are designed to process sequential data with temporal dependencies (Azzouni, 2017). This makes them suitable for tasks like anomaly detection in network traffic, where identifying deviations from normal behavior over time is crucial. They can handle long-term dependencies within network traffic data, enabling them to detect complex attack patterns that unfold over time.

Nazir et al. (2023) discuss the growing weaknesses and attacks on the IoT environment. With the IoT landscape projected to connect over 20 billion devices by 2024, securing this interconnected system is increasingly critical. The researchers introduce a new method using Deep learning methods using Recurrent Neural Networks (RNNs) for detecting intrusions in Internet of Things environments. They utilized the NSL-KDD dataset for training and testing their model. The proposed RNN model achieves a significant accuracy of 87% in identifying intrusions. This research highlights the importance of using machine learning and deep learning techniques to enhance security in the evolving IoT ecosystem. Additionally, the authors plan to investigate optimization algorithms in future work to improve the detection accuracy of their model.

Convolutional Neural Networks (CNNs): CNNs are primarily used for processing image and video data, excelling at identifying spatial patterns. However, they can be

adapted for network traffic analysis, where they can identify patterns within fixed-size blocks of network traffic data (Demmes et al., 2023).

Khan et al. (2019) introduces a novel Convolutional Neural Network of NIDS, which integrates the CNN algorithm with the softmax algorithm. The model's intrusion detection method comprises three main steps. Initially, data preprocessing involves converting symbolic data to numeric data followed by data normalization. The second step involves using a CNN model with three hidden layers for feature extraction and data training. Finally, the softmax classifier is employed to obtain the classification results. The proposed Network Intrusion Detection System (NIDS) is evaluated using the KDD99 dataset. Experimental results demonstrate that the model outperforms Support Vector Machines and Deep Belief Networks in terms of intrusion detection accuracy. This accuracy improves with an increase in the number of epochs. The study suggests that future evaluations of the model should Evaluate its performance against the latest state-of-the-art network intrusion techniques and its suitability for IoT and cloud computing environments, which are characterized by large-scale data and sophisticated attacks.

Generative Adversarial Networks (GANs): GANs are composed of two competing models: a discriminator and a generator. The discriminator attempts to discern between produced and real data, while the generator produces artificial data that mimics actual network traffic (Navidan et al., 2021). This adversarial training process enables the discriminator to learn intricate features of normal network traffic, which can help in detecting anomalies during real-world deployments.

Autoencoders: Autoencoders are unsupervised learning models that can be used for anomaly detection. They are trained to reconstruct normal network traffic data (Song et al., 2021). Significant deviations from the reconstructed data during the testing phase might indicate an anomaly or potential attack.

Gurung et al. (2019) propose a deep learning-based IDS that leverages a sparse auto-encoder and logistic regression to address network intrusion issues. The model uses a sparse auto-encoder with a sparsity constraint for training, followed by logistic regression in the classification phase, resulting in binary classification of users as normal or intruders. The model is trained and evaluated using the NSL-KDD dataset and demonstrates higher accuracy rates in contrast to signature-based intrusion detection methods, with fewer false positives and false negatives. While this technique is particularly suitable for real-time network monitoring servers, there is no prove yet of its application in real-world scenarios.

Chawla et al. (2019) presented a method for HIDS using Recurrent Neural Networks (RNNs) that rely on system call sequences. Their system is designed to detect security breaches by identifying suspicious or anomalous behavior in vast amounts of data. One of the significant contributions of the system is the inclusion of Gated Recurrent Units (GRUs) in place of traditional LSTM networks to improve computation efficiency and training time. Additionally, the authors combined GRUs with Convolutional Neural Networks (CNNs) to improve anomaly detection. The authors applied the ADFA dataset to verify the effectiveness of their approach, which demonstrated that the CNN/GRU model is about ten times quicker than LSTM, thanks to faster convergence during training. The authors got a 99% true positive and a 61% of false negative with

their proposed model. However, to achieve better accuracy, augmenting the number of training samples or testing the approach on separate datasets would be necessary.

Zeyuan et al. (2019) presents a novel system for detecting anomalies in computer network traffic using recurrent neural networks (RNNs). The study addresses the critical need for an effective method to detect network anomalies and prevent intrusion. The author highlights the challenges of traditional rule-based intrusion detection systems, which often need help to adapt to new threats. The proposed system comprises three stages: preprocessing, feature extraction, and classification, to find anomalies in network traffic. The RNN model is used to model sequential patterns and classify traffic as either normal or anomalous, offering a promising solution to the challenges of network intrusion detection.

Park et al. (2021) presented a model for detecting intrusions. The model involves several steps, including vector-to-image processing, host-based preprocessing, and training, and testing the model to improve its overall performance. A Siamese-CNN was developed using a learning technique that achieved high performance with minimal data during the training and testing steps. By converting each attack sample to an image, the Siamese-CNN classifies the type of intrusion based on its resemblance ratio. Accuracy was measured using few-shot learning techniques. Researchers compared Siamese-CNN and Vanilla-CNN accuracy, F1-score, recall, and precision to assess the model. The findings indicate that the suggested Siamese-CNN model had approximately 6% higher accuracy than the Vanilla-CNN model. This suggests that the proposed approach effectively improves the accuracy of intrusion detection for well-known and new cyber-attacks. Future development of this model could involve

optimizing hyperparameter values to increase the accuracy of intrusion detection (Park et al., 2021).

Zhang et al. (2019) proposed a multi-layered approach that combined CNN and GcForest machine learning techniques to identify attacks. GcForest is a random forest method that builds decision trees in a cascade structure. The model had two parts. The first part used the GoogLeNetNP CNN algorithm to differentiate between various types of attacks in the input data. The second part employed a deep forest model to produce more attacks. Their solution's precision was improved by categorizing the anomalous classes into N-1 subclasses. The cascade principle of gcForest was used to create the second layer, but XGBoost was used instead. XGBoost is a method that constructs trees sequentially until the objective function is optimized, like a random forest. The researchers evaluated their method on two datasets, “UNSW-NB15” and “CICIDS2017,” and achieved 99.24% accuracy, surpassing the performance of singularly used algorithms (Zhang et al., 2019).

2.5 Use of Dataset in Machine Learning and Deep Learning

Several datasets have been utilized to develop and test AI-based intrusion detection systems. The KDDCUP99 dataset, originating from US Air Force network traffic and made public in 1999, was commonly used. However, due to its limitations and flaws affecting the accuracy and validity of IDS research, the NSL-KDD dataset was developed to address these issues and remove unnecessary records from the KDDCUP99 dataset (Tavallaei et al., 2009). The training set composed of 125,973 records, and the test set has 22,544 records. Another dataset is the CICIDS2017, which includes both normal and malicious network traffic, with new types of attacks such as

“Distributed Denial of Service”, “DoS”, “Botnet Web Attacks”, “Brute Force FTP”, “Brute Force SSH”, “Heartbleed”, and “Infiltration”.

The CICIDS2017 dataset was created using a complex network topology consisting of modems, routers, switches, firewalls, and nodes with various operating systems. It contains 80 network flow features extracted from the captured network traffic. The Australian Cyber Security Centre also created the UNSW-NB15 dataset, generated using the IXIA PerfectStorm tool. It includes nine types of attacks, and 49 features extracted from network logs. The UGR’16 datasets, developed in 2016 using data from the Spanish information service provider network, is another widely used dataset (Maciá-Fernández et al., 2018).

Table 2.1

Commonly Used Ids Datasets and Types of Attacks Featured

Dataset	Attack Category	year	Format	Size	Traffic type
KDDCUP 99	Probing, remote-to-local, user-to-root, distributed denial of service attacks.	1999	Other	5 MB.	Emulated
Dataset	Attack Category	year	Format	Size	Traffic type
NSL-KDD	Ddos attack, probing, remoting to the local machine, user-to-root	2009	Other	150kb	Emulated
UNSW-NB15	DoS, generic exploits, fuzzes, worms, port scanning, reconnaissane,	2015	Packet & others	2 MB.	Emulated

	backdoors, shellcode, spam				
UGR'16	Port scanning, SSH brute force, Spam, botnet, DoS	2016	Bi-directional flow	17 GB.	Real
CICIDS20 17	A cross-site-scripting botnet, SQL injection, DoS, Distributed Denial of Service attack, Heartbleed, infiltration, SSH brute force	2017	Bi-directional flow	51.3GB	Emulated
CSE-CIC- IDS-2018	Cross-site-scripting botnet attack, SQL injection, DoS attack, brute force attack, infiltration attack.	2018	Bi-directional flow		Real

Note. This shows the type of dataset commonly used in ai-based intrusion detection systems.

2.6 The Gap in the Literature

Despite the significant progress in applying AI-based intrusion detection systems, some gaps in the Literature still need to be addressed. One of these gaps is the reliance of several studies on outdated datasets. These datasets may need to be revised to represent the current network security landscape adequately; as a result, the effectiveness of the intrusion detection system may be restricted.

It is important to conduct further research using up-to-date datasets. This helps to increase the accuracy of AI-based intrusion detection systems.

Additionally, there needs to be more research on the effectiveness of AI-based systems in preventing network intrusion. Much research has focused on detecting network intrusion, but more needs to be done to prevent such attacks from occurring. Therefore, Further Research in this area could improve the overall security of networks and reduce the risk of successful attacks.

In addition to the gaps in Literature, several other challenges need to be addressed in the field of intrusion detection systems. One of these challenges is the difficulty in detecting new threats. Signature-based detection methods, which rely on rules to determine malicious network traffic, can struggle to identify new or modified threats (Shenfield et al., 2018).

Another challenge is the issue of false positives and false negatives. Adversarial AI attacks can inject malicious input into AI training data, causing the system to flag good traffic as malicious (false positive) incorrectly or allow malicious traffic to enter the system (William, 2021).

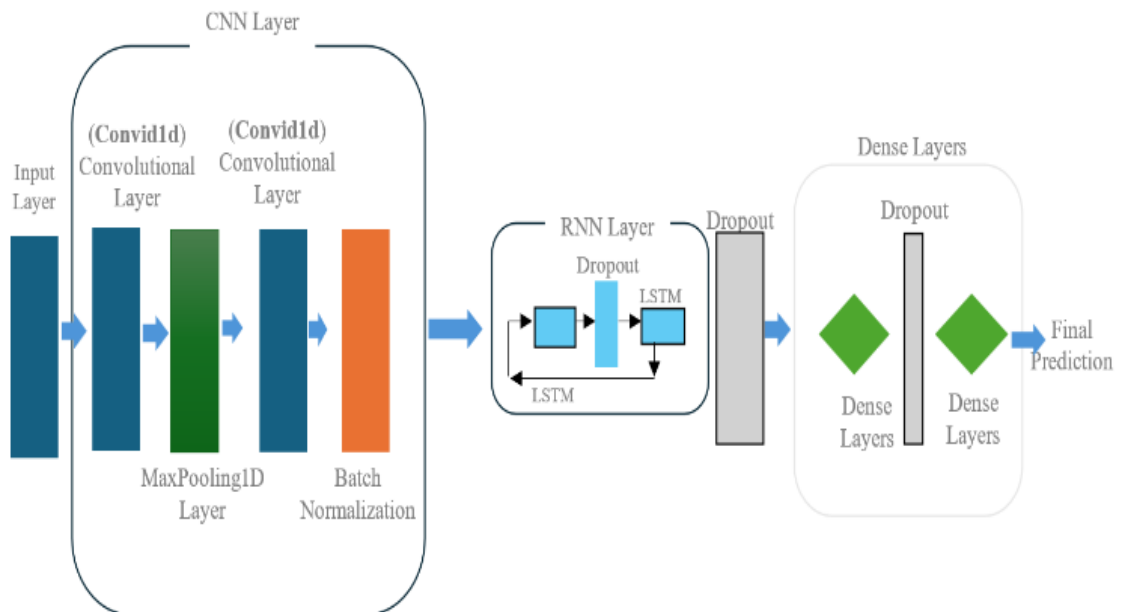
2.7 The proposed model Architecture

In this study, a hybrid deep learning architecture was developed, integrating Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) units, for network intrusion detection. This architecture leveraged the strengths of both CNNs and RNNs to enhance the accuracy and effectiveness of detecting intrusions in network traffic data. The model's input layer received raw network traffic data, which was pre-processed and normalized. The CNN component comprised multiple layers, including convolutional layers, a MaxPooling1D layer, and batch normalization, which collaboratively

extracted local patterns and hierarchical features from the data. The high-level features extracted by the CNN were then passed to the RNN component, where LSTM units captured temporal dependencies crucial for identifying network attacks that manifested over time. To prevent overfitting, dropout layers were strategically placed before the LSTM and dense layers. The dense layers further refined the features, culminating in a final prediction layer that classified the network traffic as either normal or indicative of an intrusion. This hybrid approach, by combining the spatial feature extraction capabilities of CNNs with the temporal sequence handling strengths of RNNs, demonstrated superior performance in accurately identifying various types of network intrusions. The architecture's design allowed for scalability and adaptability, making it suitable for real-time network intrusion detection systems. Fig 4 presents the model architecture.

Figure 2.4

The proposed model Architecture



2.8 Literature Review Summary

This research examines a range of machine learning techniques utilized in intrusion detection systems, highlighting the application of deep learning methods for identifying and preventing network intrusions. The primary goal is to enhance the effectiveness and accuracy of network intrusion detection systems.

CHAPTER THREE

METHODOLOGY

Intrusion detection systems have been the subject of much research works. In the precedent chapter, numerous deep learning techniques used in intrusion detection tasks using dedicated datasets for cybersecurity were discussed. The methodology encompasses several key steps, including dataset selection, tools and environment setup, model development, data collection, data preprocessing, and model evaluation.

3.1 Dataset

In this research, the CSE-CIC-IDS-2018 datasets were used to train and assess the model effectively. This dataset had been chosen due to its recentness, realistic representation of network logs, and a wide range of network attacks and regular activities. Furthermore, it offers extensive features and labels for conducting a thorough network traffic analysis and was publicly available. The CSE-CIC-IDS-2018 datasets were developed in 2018 by the Canadian Institute of Cyber Security through meticulous intrusion traffic (Aws Data Exchange, 2018). These datasets are composed of seven categories that accurately describe various attacks

3.2 Tools and Environment

For model development, the research utilized the Azure machine learning. Azure Machine Learning is designed to assist developers and data scientists build, deploying, and managing machine learning models.

Azure Machine Learning is a Cloud-based platform where you can create a virtual computer for model development. Azure for machine learning was selected because it offers comprehensive tools and capabilities that streamline the entire machine-learning

lifecycle. You need an Azure account to use this service. The study used Python, which is a programming language with several platforms, multiple paradigms, and interpretation. Due to its versatility and extensive range of pre-installed libraries, it is also widely utilized and well-liked in the fields of machine learning and artificial intelligence. These libraries include Pandas, Numpy, Scikit-Learn...etc. Pandas and NumPy are used for data manipulation; scikit-learn allows to experiment with different machine learning techniques and pre-defined data analysis algorithms quickly and easily. TensorFlow and Keras's frameworks were chosen to implement the proposed deep learning methods. TensorFlow is a Google open-source deep learning toolkit used for Deep Learning model architectures by performing difficult numerical calculations and other activities. It offers the advantage of being able to easily deploy calculations on various platforms, including CPUs and GPUs, and was developed to enable rapid experimentation. Keras is a high-level API created to facilitate the creation and training of deep-learning models using Python. It was created to enable quick experimentation. Among these advantages:

NumPy: This is a fundamental library for numerical computing in Python. This library helps in data preprocessing and handle data in the form of arrays. NumPy allows you to perform various operations like reshaping, slicing, indexing, and aggregations on datasets. NumPy is also utilized for linear algebra operations, generating random numbers, and computing mathematical functions, which are crucial in machine learning algorithms.

Pandas is a library in Python that offers efficient and user-friendly tools for data analysis and structures. It is widely used in machine learning for data manipulation, cleaning, and exploration tasks.

Matplotlib: This data visualization library in Python provides a flexible and comprehensive set of plotting functions. This library was used to visualize data, explore patterns, and understand the results of the proposed model. With Matplotlib, you can generate a diverse range of plots like line plots, bar plots, histograms, scatter plots, heat maps, and more.

Scikit-learn provides various preprocessing approaches that could be applied to the input data before training a deep learning model. These techniques include scaling features, handling values that are missing, encoding categorical variables, and reducing dimensionality. Scikit-learn was used to ensure the input data is prepared and standardized before feeding it to the model.

3.3 Model development

This research used deep learning techniques for model development, utilizing a CNN and a RNN algorithm. These algorithms were selected because they could perform better results than ML Algorithms in detecting intrusions in the network. The CNN used pooling layers and convolutional filters to extract features from images and other kinds of data. The RNN was able to analyze sequential data and capture temporal dependencies. It could identify various network incursions by categorizing the network monitoring data into distinct groups. By taking elements from network monitoring data and categorizing distinct sorts of intrusions, it was able to identify various forms of network intrusions.

Thus, by combining the CNN with the RNN, a more potent and adaptable model was produced that could handle temporal and spatial information in network data. They improved their feature extraction and categorization capability by learning from each

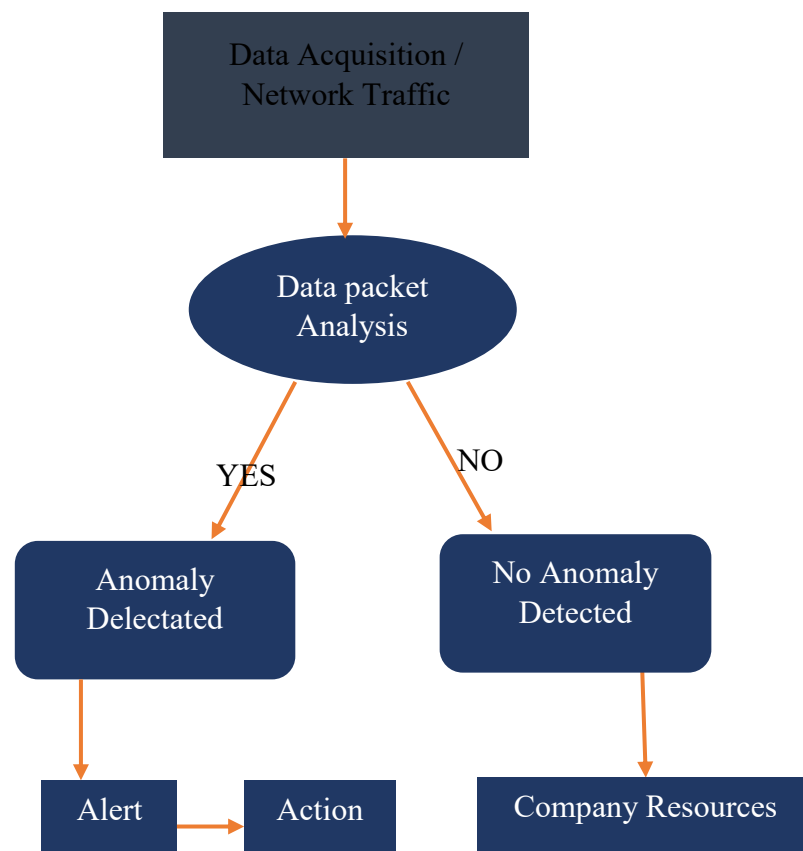
other. Moreover, deep learning algorithms demonstrated the capacity to identify novel and new attacks, as they could learn from complex data. Furthermore, the implementation of deep learning techniques demonstrated encouraging outcomes in enhancing the precision and efficiency of intrusion detection systems, making them a valuable tool in network security.

3.3.1 Model development Architecture

The model was able to detect network intrusion and sent an alert to the network administrator so they could take appropriate action. The proposed model used realistic datasets that were up to date. Figure 4 shows the model development architecture.

Figure 3.1

Architecture for model development



To build this model, the research focused on the following steps – Data collection, Data preparation, determining the model features and training the model, and finally, Evaluating the model performance.

3.3.2 Data collection

In this research, the CSE-CIC-IDS-2018 dataset was collected from the official website of the Canadian Institute for Cybersecurity (CIC). They provided access to the dataset on their official website. (University of New Brunswick, 2018)

3.3.3 Data preprocessing

The data was preprocessed to make it suitable for the AI-based model. This involved removing any irrelevant or redundant features. Raw data usually contains noisy, missing values, and unsuitable formats that could not be utilized directly for machine learning models. Furthermore, the datasets utilized in intrusion detection were highly dimensional, making it more challenging to acquire information during training. Data preprocessing was important in cleaning and preparing data for the model. It enhanced the efficiency and accuracy of the model. An illustration of the data preprocessing procedure is shown in Figure 5.

Data Preprocessing



3.3.5 Feature encoding

One-hot encoding is a technique for using binary vectors to represent categorical information. In the original functionality, it creates new binary columns for every category (Bobbitt, 2022).

For every unique category in the original feature, a new binary column is generated. If an observation belongs to a specific category, the corresponding binary column is assigned a value of 1; otherwise, it retains a value of 0.

Example: Suppose we have a categorical feature called “Team” with three unique values: A, B, and C. Using one-hot encoding, we create three new columns: Team_A, Team_B, and Team_C. If A was the initial value in the "Team" column, the value in the "Team_A" column is 1; if not, it is 0. Similarly, Team_B and Team_C columns represent the other categories.

Table 3.1

One-hot encoder

Team	Points	One-hot encoder			Points
		Team_A	Team_B	Team_C	
A	25	1	0	0	25
A	30	1	0	0	30
C	25	0	0	1	25
B	20	0	1	0	20
A	27	1	0	0	27
B	31	0	1	0	31
C	28	0	0	1	28

Use Case: One-hot encoding is commonly used when there is no inherent order or ranking among the categories. It’s suitable for nominal data.

Label encoding gives each label a distinct integer to each category in a categorical feature. Each category is mapped to an integer value (Bobbitt, 2022). The order of mapping is arbitrary and does not imply any ordinal relationship between categories.

Example: Suppose we have the same “Team” feature with values A, B, and C.

Table 3.2

label encoding

		label encoding	
Team	Points	Team	Points
A	25	0	25
A	30	0	30
C	25	2	25
B	20	1	20
A	27	0	27
B	31	1	31
C	28	2	28

Use Case: Label encoding is suitable for ordinal data (where there is a meaningful order) or when working with a significant number of categories.

In most cases, one-hot encoding is the preferred method because label encoding may mistakenly imply a ranking between values, even when no such ranking exists. For instance, label encoding could make it seem like team C is bigger than team's B and A, which might not be accurate for nominal data. It is crucial to consider the nature of the data and the specific context when choosing an encoding method.

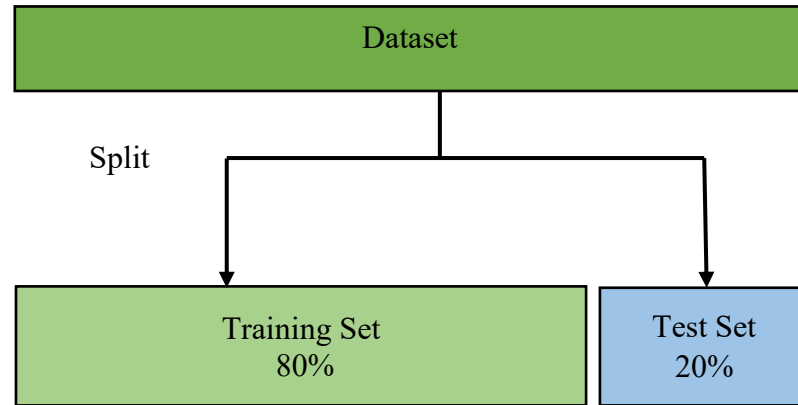
3.3.6 Splitting the Dataset

To improve the performance of the ML model, the data was split into two categories: A training set and a test set, with the training set making up 80% of the total data, while the test set had 20%, as depicted in Figure 6. This was a crucial step in data preprocessing as it helped the model understand the correlations between the data. Using a completely different dataset to test the trained model could result in difficulty comprehending the correlations between the models. Therefore, even if the model had high training accuracy, it might not perform well with a new dataset. The goal was

always to develop a machine learning model that performed well with both the training and test datasets.

Figure 3.3

split datasets



3.3.7 Feature scaling

Feature scaling was performed to scale the dataset's features to make sure all features were in the same range and scale. This prevented any variable from dominating another. In this study the standardization technique was used, as this method was effective for CSE-CIC-IDS-2018 dataset.

Here are the two common techniques for feature scaling for machine learning:

Min-Max Scaling (Normalization)

$$X' = \frac{X - \min(x)}{\max(x) - \min(x)} \quad (1)$$

The values of the features are scaled down between 0 to 1. Here, the new value represents the scaled value and 'x' is the original value.

Standardization (Z-score normalization):

$$X' = \frac{X - \text{mean}(x)}{a} \quad (2)$$

A standard normal distribution with a mean of 0 and a standard deviation of 1 is created by transforming all features. X represents the new value, while the original value is represented by mean. The mean is represented by (x) , and the standard deviation is represented by (a) .

3.4 Model evaluation

In that step, the performance of the suggested model was evaluated using a testing set of data. The evaluation metrics included accuracy, recall, F1-score, and precision, which measured how well the model correctly classified the network traffic as usual or malicious. A confusion matrix that displayed the quantity of true positives, false positives, true negatives, and false negatives in the model's output was also included in the evaluation.

Accuracy (Acc)

These metrics define the proportion of the correct predicted number over the total predicted number. Accuracy is defined as follows:

$$\text{Accuracy} = \frac{\text{Number of correct prediction}}{\text{Total number of predictions}} \quad (3)$$

Precision (PPV)

The precision determines the ratio of the value predicted as positive that was positive. Accuracy is defined as follows:

$$\text{precision} = \frac{\text{True positive}}{(\text{True positive} + \text{False positive})} \quad (4)$$

Recall (TPR.)

recall is like precision, but this metric calculates the proportion of really positive that was discovered as negative. It can be calculated as a True Positive or an accurate forecast. of the total number of positives, predicted as a true positive, or a false negative. The recall is defined as follows:

$$\text{Recall} = \frac{\text{True Positive}}{(\text{True positive} + \text{False Negative})} \quad (5)$$

F1-score (F1)

The following formula defines the F1-score.

$$\text{F1 - Score} = 2 * \frac{\text{Precision} * \text{recall}}{\text{precision} + \text{recall}} \quad (6)$$

CHAPTER FOUR

DISCUSSION AND RESULT

This section presents the findings for the approaches outlined in the previous section. An AI-based intrusion detection model was trained and validated utilizing the CSE-CIC-IDS2018 dataset. The performance, accuracy and F1Score were calculated on a different subset designated as the testing dataset.

4.1 Exploratory Data Analysis (EDA)

An in-depth Exploratory Data Analysis (EDA) was carried out to gain insights into the CSE-CIC-IDS2018 dataset. Exploratory Data Analysis (EDA) is a technique of analyzing data that is utilized to recognize overall trends within the data, which may encompass outliers and unforeseen data characteristics (Wathen et al., 2015). The EDA is divided into data Visualization and Data Understanding.

The CSE-CIC-IDS2018 dataset, which was provided by the Canadian Institute for Cybersecurity (CIC), was used in this research. It consists of real network flow data with several types of the most recent and widespread Network attacks such as DDOS attacks, Brute Force, Bot, Infiltration, and SQL Injection. These data were presented in more compact formats that primarily carry meta-data regarding network connections, with each network flow grouping all packets that share specific attributes in a time frame and excluding payload. The dataset includes two types of data: raw PCAP data and CSV data.

The structured and labelled CSV files with over 80 features were provided by the University of New Brunswick. The researcher used a CICFlowMeter-V3 traffic analyzer to derive the characteristics from the PCAP files (Sharafaldin et al., 2019). CICFlowMeter is a flexible Java-based network traffic flow generator that lets users compute certain features, add more metrics,

and control flow timeout durations. With the first packet determining the direction of both forward (from source to destination) and backward (from destination to source) traffic, it generates Bidirectional Flows (Biflow). As a result, it can calculate 83 distinct statistical properties, such as duration, packet count, byte count, packet length, and more (University of New Brunswick, 2018). The downloaded dataset comprised of seven different CSV datasets. In this study, we combined all the CSV data into a single dataset that had all of the features required for the model.

The CIC-IDS2018 dataset includes 7 different types of attacks, the most up to date. The complete set of data contains 2,476,970 instances, of which 1,448,511 are attacks. The most prevalent is the DDOS attack-HOIC with 686,012 instances, followed by Bot activities and FTP-BruteForce attacks with 286,191 and 193,360 instances respectively. SSH-Bruteforce and Infiltration also appear significantly. There are 1,028,459 instances of benign (normal) network traffic. The number of instances for each type of network attack is indicated in Table 4.2

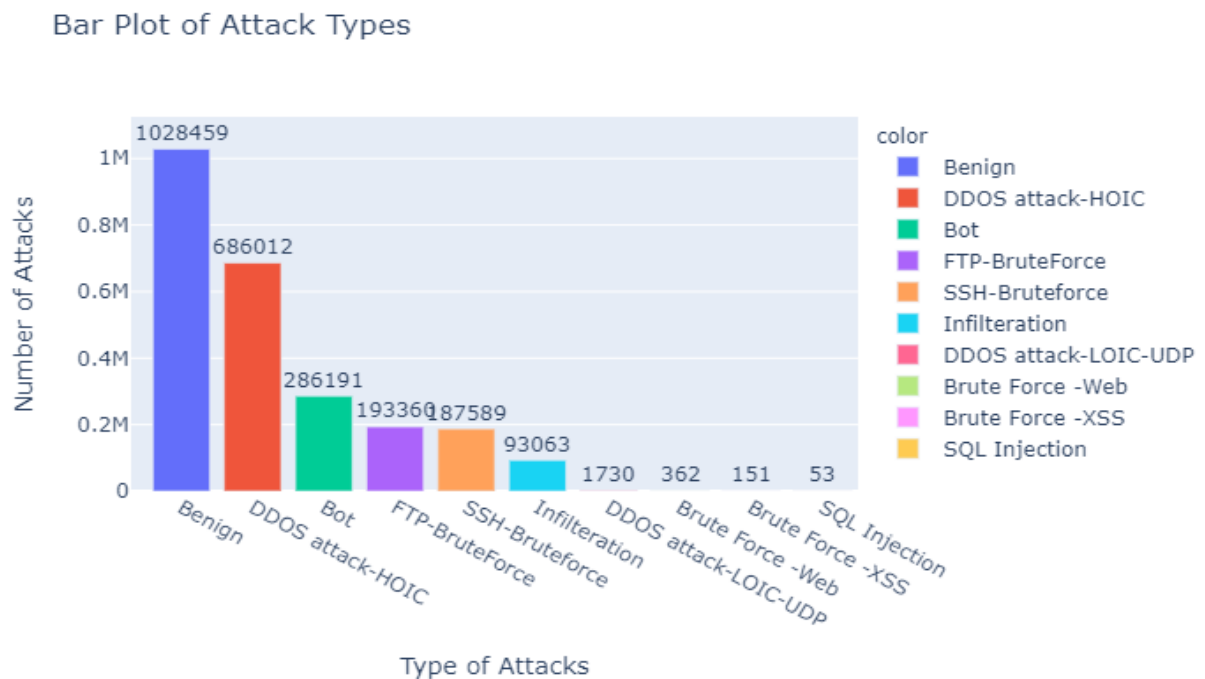
Table 4.1

CSE-CIC-IDS2018: Number of instances for each type of attack

Class Label	Number of instances
Benign (Normal network Traffic.)	1028459
DDOS attack_HOIC	686012
Bot	286191
Brute_Force_FTP	193360
Bruteforce_SSH	187589
Infiltration	93063
Attack-LOIC-UDP DDOS	1730
Brute Force -Web	362
Brute Force _XSS	151
SQL Injection	53

Figure 4.1

CSE-CIC-IDS2018: Number of instances for each type of attack



4.2 Data Preprocessing

Data preprocessing is a crucial data mining technique that ensures better outcomes from the data analysis process. It not only helps in improving the quality of the data but also helps in extracting meaningful and useful information from the data. It prepares the data so that it becomes easy to work with and the quality of the data is maintained and improved wherever possible. The dataset is preprocessed before being fed into the deep neural network.

The dataset was initially filtered to eliminate any redundant rows that represent class instances. Then, an analysis was performed to find any 'NaN' (Not A Number) or 'INF' (Infinite Value) values, which are presented as missing values. The performance of the final models is directly and severely impacted by how poorly deep learning or machine learning algorithms handle these values in general. It appears that the data selected has columns with missing values for the Flow Bytes column; to keep this feature, the missing value was filled with a specific value

like the mean. This technique ensures the integrity of the data while maintaining the relevance of the 'Flow Bytes' feature.

4.2.1 Feature selection

The method of selecting features involves determining the most important attributes for effective classification or prediction (Venkatesh and Anuradha, 2019). It helps decrease resource requirements such as storage and computation and can increase the performance of machine learning algorithms with faster training, reduced overfitting, and improved prediction accuracy. Although it may appear to lead to information loss, this isn't the case when dealing with redundant or irrelevant information. Redundant features contain the same information as other attributes or are derived from combinations of other features. There are three main classes of feature selection techniques. Filtering methods involve pre-processing features to eliminate those that may not be useful for the model. One method, for example, would be to figure out the mutual information or correlation between each characteristic and the response variable, and then exclude any features that don't match a predetermined cutoff.

Embedded methods include choosing features for the model during training. For instance, a decision tree inherently conducts feature selection by choosing a feature to split the tree at each training step. An additional illustration is the ℓ_1 regularize, which may be incorporated into any linear model's training objective.

Wrapper approaches: Although they require more resources, these methods let you experiment with feature subsets, preventing the accidental removal of potentially useful features. The model is viewed as a black box by the wrapper approach, which offers a quality score for a suggested feature subset. Another method involves iterative refinement of the subset. The descriptive statistics, which provide a summary of the dataset's dispersion and distribution, revealed the presence of empty columns (i.e., columns where all values are zero). These

columns include “Bwd PSH Flags”, “Bwd URG Flags”, “Fwd Byts/b Avg”, “Fwd Pkts/b Avg”, “Fwd Blk Rate Avg”, “Bwd Byts/b Avg”, “Bwd Pkts/b Avg”, “Bwd Blk Rate Avg”, “Flow ID”, “Source IP”, “Destination IP”, “Timestamp”, “Inbound”

These features do not contribute any discriminatory information that could help differentiate between the attack classes. In fact, they could potentially lead to suboptimal results. Consequently, these columns were removed from the dataset.

In data preprocessing, it’s crucial to remove features (columns) that don’t contribute to the model’s predictive power. This is because machine learning algorithms might be overfitting to these non-informative features, leading to poor generalization performance. In this case, the columns that always have a value of zero don’t provide any information that can help the model distinguish between different attack classes. Therefore, they were removed to improve the model’s performance.

Features like ‘IP Address’ and ‘Timestamp’ were deemed to be related to connection information and not representative of the characteristics of network attacks. Given that network attacks can be initiated by any machine, at any time, against any target machine, features like ‘Flow ID’, ‘Source IP’, ‘Destination IP’, ‘Timestamp’, and ‘Inbound’ were also eliminated. This was done to ensure that only the features pertaining to network traffic were retained. The remaining features, which are the focus of this study, are listed in Table 5.

Table 4.2

The set of characteristics used for the model

Features	Type	Features	Type
Dst -Port	Int_64	Pkt Len Max	int64
Protocol	Int_65	Pkt Len Mean	float64
Flow-Duration	Int_66	Pkt Len Std	float64
Tot Fwd-Pkts	Int_67	Pkt Len Var	float64

Tot Bwd-Pkts	Int_68	FIN Flag Cnt	int64
Features	Type	Features	Type
Tot_ Len Fwd Pkts	Int_69	SYN Flag-Cnt	Int_64
Tot_ Len Bwd Pkts	Float_64	RST Flag-Cnt	Int_64
Fwd_ Pkt Len Max	Int_64	PSH Flag-Cnt	Int_64
Fwd_ Pkt Len Min	Int_64	ACK Flag-Cnt	Int_64
Fwd_ Pkt Len Mean	Float_64	URG Flag-Cnt	Int_64
Fwd_ Pkt Len Std	Float_64	CWE Flag-Count	Int_64
Bwd_ Pkt Len Max	Int_64	ECE Flag-Cnt	Int_64
Bwd_ Pkt Len Min	Int_64	Down/Up Ratio	Int_64
Bwd_ Pkt Len Mean	Float_64	Pkt_Size Avg	Float_64
Bwd_ Pkt Len Std	Float_64	Fwd_Seg Size Avg	Float_64
Flow Byts/s	Float_64	Bwd_Seg Size Avg	Float_64
Flow Pkts/s	Float_64	Subflow_Fwd Pkts	Int_64
Flow IA_ Mean	Float_64	Subflow_Fwd Byts	Int_64
Flow IAT_ Std	Float_64	Subflow_Bwd Pkts	Int_64
Flow IAT_ Max	Float_64	Subflow_Bwd Byts	Int_64
Flow IAT_ Min	Float_64	Init Fwd_Win Byts	Int_64
Fwd IAT_ Tot	Float_64	Init Bwd_Win Byts	Int_64
Fwd IAT_ Mean	Float_64	Fwd Act_Data Pkts	Int_64
Fwd IAT_ Std	Float_64	Fwd Seg_Size Min	Int_64
Fwd IAT_ Max	Float_64	Active Mean	Float_64
Fwd IAT_ Min	Float_64	Active Std	Float_64
Bwd IAT_ Tot	Float_64	Active Max	Float_64
Bwd IAT_ Mean	Float_64	Active Min	float64
Bwd IAT_ Std	Float_64	Idle Mean	Float_64
Bwd IAT_ Max	Float_64	Idle Std	Float_64
Bwd IAT_ Min	Float_64	Idle Max	Float_64
Fwd PSH_ Flags	Int_64	Idle Min	Float_64
Fwd URG_ Flags	Int_64	Label	object
Fwd Header_ Len	Int_64	Bwd_Pkts/s	Float_64
Bwd Header_ Len	Int_64	Pkt_ Len Min	Int_64
Fwd Pkts/s	Float_64		

4.2.2 Feature encoding

The dataset contains several categorical features that require encoding. The 'Flow Packets/s' column, for instance, was transformed into a numeric column. The 'Label' column, which signifies the class of each instance, was transformed using a widely used technique known as 'One-Hot Encoding'.

This method converts each category within the rows into its own distinct column. A value of 1 in this column indicates that the instance belongs to that particular class (true), while a value of 0 signifies that it does not (false).

'One-Hot Encoding' is a process for converting categorical information into a format that machine learning algorithms can use to increase their efficiency. In this specific context, each unique category value (class) in the 'Label' column is turned into a new column in the encoded dataset. For each record, the column corresponding to its 'Label' value is set to 1 (indicating true), while all other generated columns are set to 0 (indicating false). This allows the model to understand and use the categorical data without any loss of information.

4.2.3 Feature scaling

Standardization is a critical aspect of feature scaling, which involves transforming the input data so that each feature has a mean of zero and a standard deviation of one. This process improves the model-building phase by accelerating convergence during training and potentially introducing a regularization effect. Standardization ensures that the influence of varying feature scales is mitigated, enhancing the accuracy of data analysis and machine learning models (Alshaher, 2021).

Standardization is performed by calculating the z-score of each feature, which is given by the formula:

$$z = \frac{x - \mu}{\sigma} \quad (7)$$

where x represents the initial value, μ denotes the feature's mean, and σ denotes its standard deviation. The features are guaranteed to have unit variance and to be centered around zero thanks to this modification

4.2.4 Data Splitting

ML algorithms learn patterns from data and use them to identify unseen data. (Deshpande et al., 2018) It is important to evaluate the performance of the models on data that they have not seen during training. Divided the dataset is important for an unbiased evaluation of prediction performance. Generally, it is sufficient to divide the dataset into three sections at random.

Train Data: This data is used to train the machine learning model and consists of input features and their corresponding output labels. The model learns patterns from this data and uses them to make predictions on new, unseen data.

Validation Data: This data is used to evaluate the performance of the model during training and to tune the model's hyperparameters. It helps prevent overfitting and should be representative of the unseen data that the model encounter in the real world.

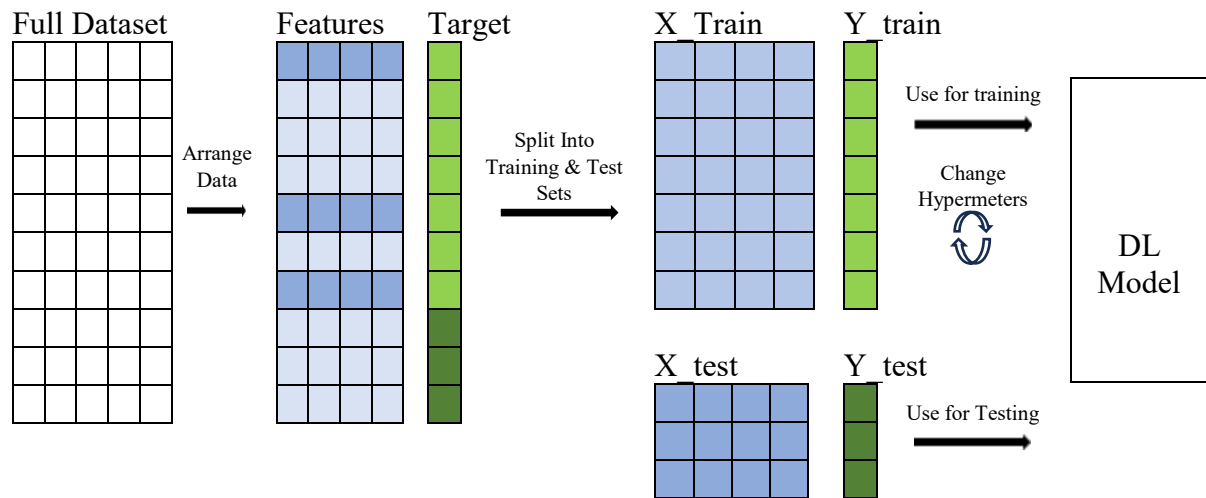
Test Data: This data is used to evaluate the final performance of the model after training and tuning. It should be separate from the training and validation data to avoid any data leakage.

The simplest data splitting strategy is random splitting, where the dataset is randomly divided into a training set and a test set. However, there is a risk of overfitting if the dataset is small or if the test set is not representative of the generalized data.

Random splitting may result in a biased dataset if the dataset is imbalanced. To avoid biases, a stratified random splitting strategy was used, which ensures that the proportion of each class is the same in both the training and test sets.

Table 4.3

Data split



4.3 Model Architecture

In this study, a hybrid model was developed by integrating CNN and RNN. Each algorithm was trained independently before being integrated into a unified model. The approach is summarized by the following key points:

The input layers are dimensioned to match the feature count of the input vector for each respective model. The Rectified Linear Unit activation function was selected after testing alternatives like hyperbolic tangent and sigmoid, with the Rectified Linear Unit consistently delivering superior performance.

For the output layers, which align with the class count, the SoftMax activation function was employed for multi-class classification. This function assigns a probability distribution across neurons, summing to one, enabling the identification of the predicted class by the neuron with the highest probability.

To counteract overfitting, the dropout technique was applied, which randomly excludes a portion of neurons from a layer during training to foster a model that generalizes well.

The loss functions chosen were categorical-cross-entropy for multi-class scenarios and binary-cross-entropy for binary classification tasks (normal/Attack).

The Adam optimizer with a learning rate of 0.001 was utilized, which outperformed the stochastic gradient descent (SGD) algorithm in the trials.

The difference between the model's predictions and the actual objectives is quantified by the loss function. Subsequently, the Adam optimizer guides the adjustment of the neural network's weights to minimize this loss, thereby expediting convergence and enhancing the accuracy of predictions with minimal error.

4.3.1 Architecture Of Convolution Neural Network Model

In the research a convolution neural network model was designed as a sequential neural network with several layers as shown in Figure 4.2, each contributing to the overall feature extraction and prediction process. Let's break down the architecture step by step:

Convolutional Layers: The Model begins with a `conv1d` layer, the workhorse of CNNs. This layer performs one-dimensional convolutions, sliding a filter across the input data to extract features. The output shape of `(None, 80, 64)` indicates that for each position of the filter, 64 features are extracted, resulting in 80 such feature sets. With 256 parameters, this layer learns the optimal filter values that highlight important characteristics in the data.

Following the `conv1d` layer is the `max_pooling1d` layer, which serves as a down-sampling mechanism. It takes the maximum value over a window of data points, effectively reducing the data's dimensionality to `(None, 40, 64)` and allowing the model to focus on the most prominent features while reducing computational load.

Next, `batch_normalization` is applied to standardize the inputs to the subsequent layer, ensuring that the network trains efficiently and effectively. A dropout layer comes next, randomly setting

a portion of the input units to 0 at each update during the training phase, preventing overfitting by ensuring that the model does not rely too heavily on any individual node. The second conv1d_1 layer, with an output shape of (None, 40, 128) and 32,896 parameters, continues the feature extraction process. It doubles the number of features extracted compared to the first convolutional layer, allowing the model to build a more complex and nuanced understanding of the data.

Pooling and Normalization, the max_pooling1d_1 layer again reduces the spatial size of the representation, halving the dimensionality to (None, 20, 128), which simplifies the network and reduces the number of parameters. This is paired with another batch_normalization_1 and dropout_1 layer, reinforcing the model's robustness and generalization capabilities.

Flattening and Dense Layers; the flatten layer is a pivotal point in the architecture. It transforms the two-dimensional feature maps into a one-dimensional vector of 2,560 elements, preparing the data for the final classification stage.

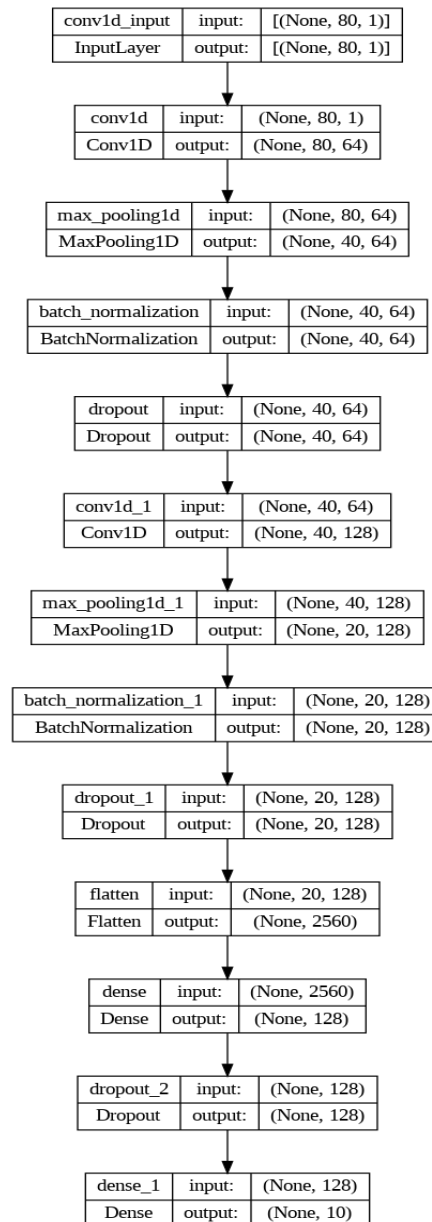
The dense layer is a fully connected layer with 128 units and 327,808 parameters. It takes the flattened data and learns complex patterns, acting as a decision-making body of the network. Another dropout_2 layer follows, ensuring that the dense layer does not overfit to the training data.

The final dense_1 layer, with 10 units and 1,290 parameters, is the culmination of the model's learning process. It outputs a 10-element vector representing the probability distribution over 10 classes, using the SoftMax activation function to normalize the output. The entire model has 92,234 parameters (approximately 360.29 KB). Among these, 91,850 parameters are trainable, meaning they are updated during training. The remaining 384 parameters are non-trainable, likely related to batch normalization. In summary, this CNN model processes input data

through convolutional layers, applies batch normalization, reduces spatial dimensions with max pooling, and finally predicts class probabilities using fully connected layers.

Figure 4.2

The architecture of Convolution Neural Network Model



4.3.2 Exploring the Architecture of a Recurrent Neural Network

The Recurrent Neural Network (RNN) presented here, uses Long Short-Term Memory (LSTM) layers to capture temporal dependencies within the data. The architecture, shown in

Figure 1, consists of several layers, each contributing to the model's overall functionality. Let's examine each layer

The network started with an Input Layer, configured to accept sequential data. Its shape is (None, 80, 1), indicating that it processes sequences with 80-time steps, each containing a single feature. This structure allows the model to handle varying amounts of data efficiently.

LSTM Layers Following the Input Layer are multiple LSTM layers, namely lstm_2, lstm_3, and lstm_4. LSTM layers are adept at learning long-term dependencies, making them ideal for time-series data like network traffic. The first LSTM layer outputs 64 features for each time step, enhancing the model's ability to discern complex patterns. The second LSTM Layer has an increased output shape of (None, 128). It aggregates information from previous layers, potentially capturing more complex features.

Dense Layers, the architecture concludes with a series of Dense layers, dense_1, dense_2, and an unnamed Dense layer, culminating in an output shape of [None, 10]. These fully connected layers consolidate the learned features and perform the final classification into one of ten possible categories, likely representing various intrusion types.

The core element of neural networks is the dense layer, sometimes referred to as the completely linked layer. Every neuron in a dense layer is linked to every other neuron in the layer before it which allows the layer to learn a global pattern in the input data. The dense layer's output is computed as a weighted sum of the inputs, plus a bias term.

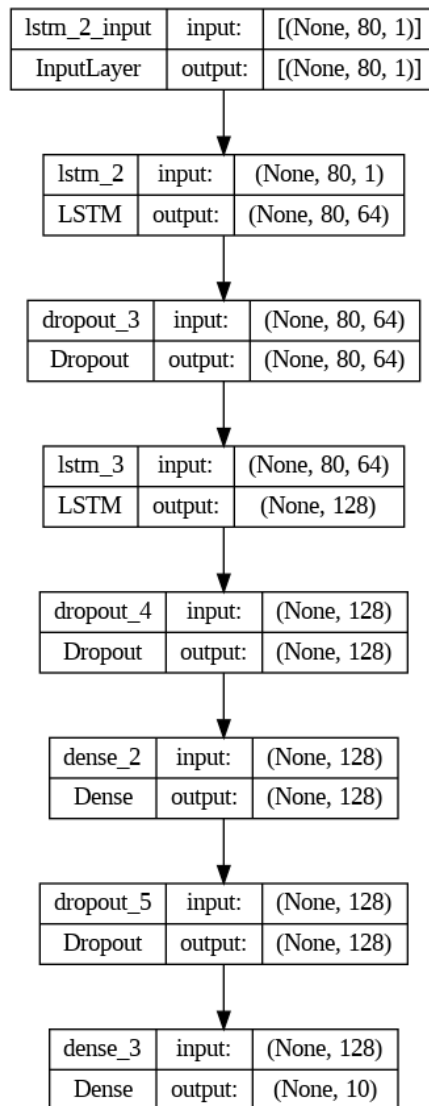
The architecture incorporates Dropout at each layer that serves as a regularization technique to prevent overfitting. They randomly set a fraction of input units to zero during training. By maintaining the output shape consistent with the preceding LSTM layers, dropout ensures robustness and generalization.

Dropout layers are a crucial component in neural network architectures, particularly for preventing overfitting during the model training process. When a model learns the training set too thoroughly—including the noise and outliers—it is said to be overfitting. This can have a detrimental effect on the model's performance on unobserved data. Dropout addresses this issue by randomly “dropping out” a proportion of the neurons in the network during training, effectively thinning out the network temporarily. This randomness forces the remaining neurons to adapt and learn more robust features that are not dependent on the presence of neurons. As a result, the network becomes less sensitive to specific weights, leading to a more generalized model that performs better on new, unseen data.

This architecture harmoniously combines recurrent (LSTM) and feedforward (Dense) layers, demonstrating its versatility in handling sequential data.

Figure 4.3

Architecture of a Recurrent Neural Network



4.3.3 Exploring the Architecture of a Hybrid Network

The sequential_2 model is a sophisticated hybrid neural network that integrates the strengths of CNNs with LSTMs to process and analyze network traffic data for potential threats.

The model begins with a Conv1D layer, which applies 64 one-dimensional convolutional filters to the input data. This layer was responsible for extracting local patterns or features from the input sequence. The output shape of this layer is (None, 80, 64), indicating that for each sample

in the batch, there are 80 vectors of 64 features each. The layer has 256 parameters, which are the weights of the convolutional filters.

Following the initial convolutional layer, is a MaxPooling1D layer, which reduces the dimensionality of the data by taking the maximum value over a window of data points. This operation helped to make the representation more abstract and reduced the computational load for subsequent layers. The output shape after this layer was (None, 40, 64), halving the size of the time dimension while keeping the feature dimension constant.

Next, the model included a Batch Normalization layer, which normalizes the activations of the previous layer at each batch. This helped to stabilize the learning process and reduced the number of training epochs required for convergence. The layer has 256 parameters, which are used to scale and shift the normalized data.

The model employed Dropout layers strategically placed after each convolutional and LSTM layer. These layers randomly set a fraction of input units to 0 at each update during training time, which helped prevent overfitting.

The second Conv1D layer increases the number of filters to 128, allowing the model to learn more complex features from the data. The output shape remains the same as the previous convolutional layer, but the number of features doubles, resulting in an output shape of (None, 40, 128).

Another MaxPooling1D and Batch Normalization layer followed, further abstracting the representation, and stabilizing the training process. The output shape after pooling was (None, 20, 128), and the batch normalization layer had 512 parameters.

The model then transitioned to the LSTM part with a layer of 64 LSTM units. LSTM layers were adapted to capture long-term dependencies in sequential data, making them ideal for sequences where context matters. The output shape of this layer is (None, 20, 64).

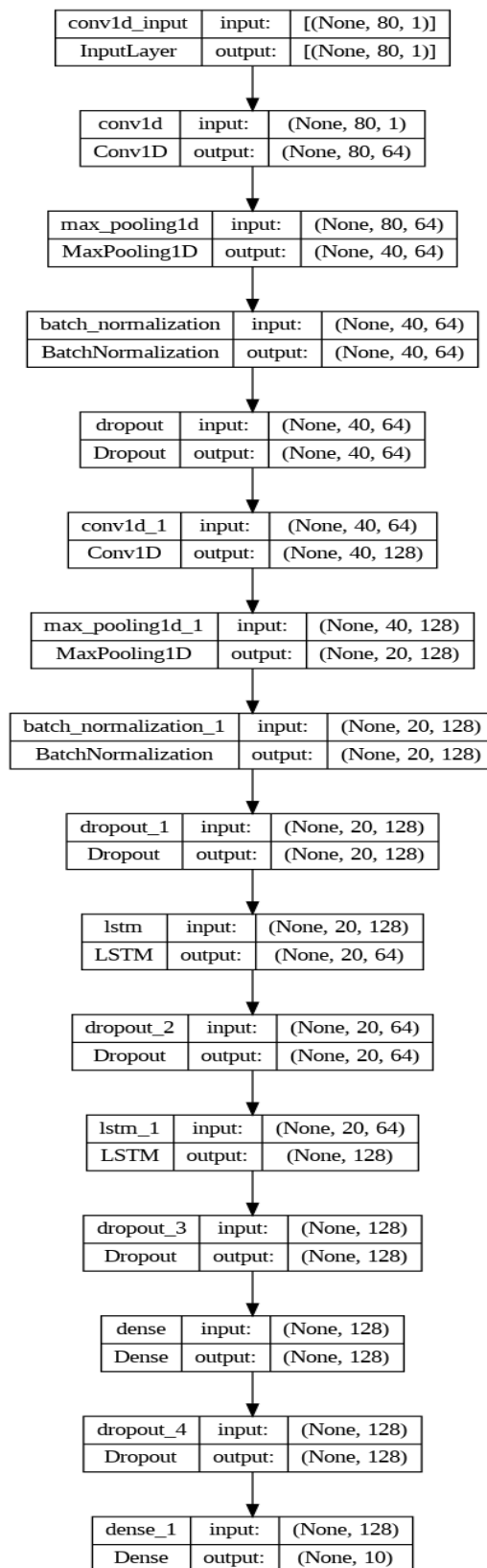
A second LSTM layer with 128 units followed, which further processes the sequential information and captures even longer-term dependencies. The output shape of this layer is (None, 128).

The model concludes with two Dense layers, which are fully connected neural network layers. The first dense layer had 128 units, and the second served as the output layer with 10 units, corresponding to the number of classes or predictions the model was designed to make. The output shape of the final layer is (None, 10).

In total, the sequential_2 model had 191,754 parameters, of which 191,370 are trainable, and 384 are non-trainable. The non-trainable parameters typically come from layers like Batch Normalization, where some parameters were used for the running mean and variance, which are learned during training but not updated during backpropagation.

Figure 4.4

Architecture of Hybrid network



4.4 Results

In this research three Deep learning models were implemented, the CNN (convolution neural network), RNN (Recurrent neural network) and a hybrid model that combines CNN and CNN. These models were trained and tested on the CIC-IDS2018 Dataset. To determine the appropriate Hyper-parameters for every model, numerous tests were conducted. These parameters comprise the elements that establish the network's architecture, such as the quantity of neurons, layers, batches, and iterations. Once we achieved a model with the minimum error rate and maximum accuracy, we then tested this model on the test subset. The results are presented in Figure 4.5.

The Dataset was divided into 2: 80% for learning and 20% for testing. The models were trained from 20 epochs. They obtained a very good accuracy of 99.86% for the CNN, 99.87% for the RNN and 99.97% when we combined both models. We noted that the two models (CNN and RNN) converge towards a minimum loss value, or they have almost the same learning and evaluation loss value. For all models, we noted that the learning and validation accuracy constantly increases from the beginning to the end, it reaches a maximum value tending towards 1. We also noted that the loss value decreases strongly during training and evaluation and reaches a minimum value tending towards 0. This means that these models learned better and make better predictions after each optimization epoch.

4.4.1 Analysis of the Models Training and Validation Performance

The performance analysis of the three models was crucial in determining their effectiveness in both learning from training data and generalizing to unseen validation data, we analyzed the training and validation curves of a CNN, RNN and The Hybrid Model to assess their behavior and ensure they are not overfitting, thereby validating their robustness and generalization capability.

Two sets of graphs are analyzed to evaluate the performance as presented in Figure 4.5. The first set illustrates the training and validation accuracy, while the second set presents the training and validation loss over a series of epochs. Both graphs are essential in understanding the learning dynamics and generalization power of the model.

The accuracy curves provide insight into how well the models were performing in classifying the data correctly. In the provided graphs, the training accuracy starts lower and increases rapidly, eventually plateauing at a value close to 1.0. in both models Similarly, the validation accuracy exhibits a swift rise, reaching a high value that remains stable across the epochs. The swift rise in both training and validation accuracy indicates that the model is learning efficiently from the data. This rapid convergence suggests that the CNN's architecture and the optimization parameters are well-suited for network detection.

The training and validation accuracy both plateau near 1.0, indicating that the models are achieving high performance on both the training and validation sets. This is a strong indicator that the models were not only learning the training data well, but they are also generalizing effectively to the validation data.

The loss curves depict the error or the deviation of the predicted values from the correct values. lower loss values are desirable as they indicate better model performance.

Both the training and validation loss curves show a consistent decrease, with the losses converging to very low values. The training loss decreases rapidly and then stabilizes, while the validation loss also shows a similar pattern, decreasing and stabilizing at a low value.

The stability of both training and validation loss at low values suggests that the models were not overfitting. Overfitting is typically characterized by a significant divergence between training and validation loss, where the validation loss begins to rise while the training loss

keeps getting smaller. This divergence indicates that the model is memorizing the training data rather than learning to generalize from it.

Overfitting occurs when a model performs well on training data but poorly on validation or test data. It is often identified by the following characteristics:

Disparity in Accuracy: In the event that the validation accuracy is substantially lower than the training accuracy, it indicates that the model is overfitting. In the provided graphs as shown in the figures 4.5, both training and validation accuracies are high and close to each other, that shows that the models were not overfeeding.

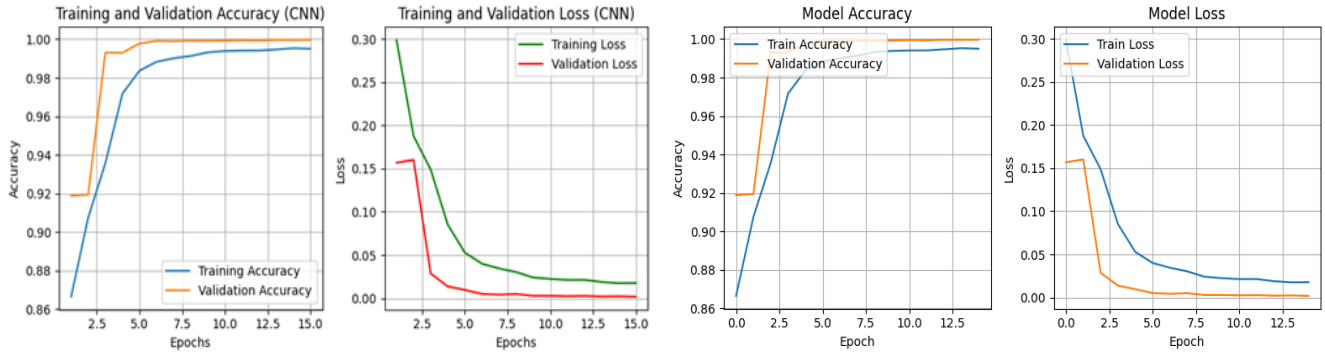
Discrepancy in Loss: A large gap between training and validation loss, with the training loss being much lower, is a classic sign of overfitting. The provided graphs in Figure 12 do not exhibit this behavior; the training and validation losses are low and closely aligned.

The analysis of the training and validation curves for the three models demonstrated that the models were not overfitting. Both the accuracy and loss graphs demonstrate that the models were learning successfully from the training data and generalizing well to the validation data. The high and stable accuracy values, along with the low and stable loss values, reinforce the model's robustness and its capability to generalize well.

Figure 4.5

Models Training and Validation Performance

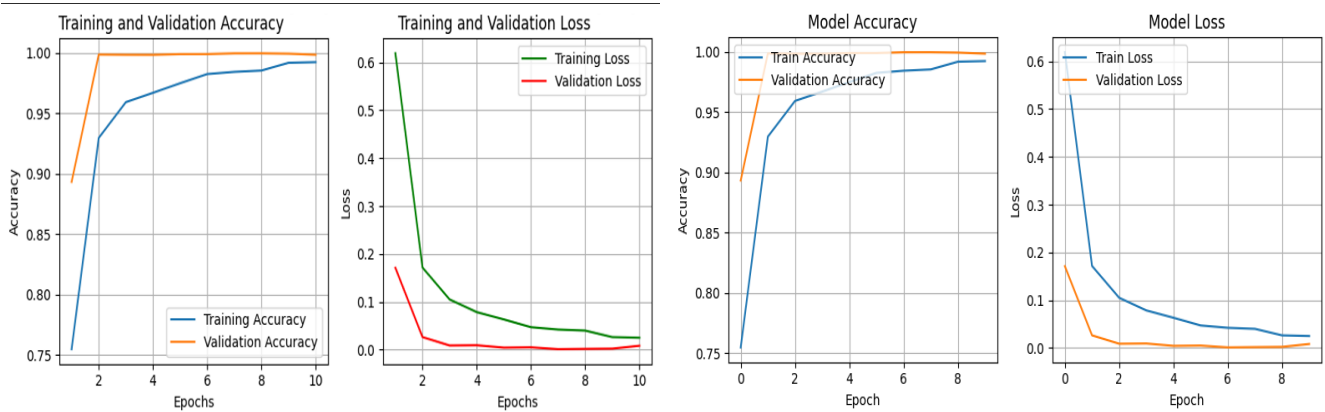
CNN MODEL



RNN MODEL



HYBRID MODEL



4.4.2 The evaluation metrics for the models

Precision (PPV): The percentage of network attacks identified as TP attacks among all predicted examples as attacks can be calculated as follows:

$$\text{precision} = \frac{TP \text{ Attack}}{(TP + FP \text{ Benign.})} \quad (8)$$

Recall (TPR): The proportion of network attacks that are correctly identified as true positives (TP) out of all actual attacks present in the dataset.

$$\text{Recall} = \frac{TP \text{ Attack}}{(True \text{ positive} + FN \text{ Attack})} \quad (9)$$

F1-score (F1): The weighted harmonic means of precision and recall.

$$F1 - \text{Score} = 2 * \frac{Precision * recall}{precision + recall} \quad (10)$$

Accuracy (Acc): The proportion of correctly predicted instances out of the total number of predicted instances.

$$\text{Accuracy} = \frac{\text{Number of correct prediction}}{\text{Total number of predictions}} \quad (11)$$

A confusion matrix is a performance measurement tool for machine learning classification (Agrawal et al., 2023). It allows for the visualization of the performance of an algorithm. The matrix itself is a two-dimensional table where the classifier model is on one axis (vertical), and the actual truth is on the other (horizontal). Each cell in the matrix represents the number of predictions made by the model, where the predicted type matches or does not match the actual type. The following are some of the experiment's performance metrics:

1. True Positive (TP) to display malicious data that has been appropriately identified as malicious.
2. To show the benign data that has been appropriately identified as benign, use True Negative (TN).
3. False Negative (FN) to display the dangerous data despite it being categorized as harmless.

4. False Positive (FP) refers to data that appears benign but is really labeled as harmful.

The confusion matrix herein presents the results of a network intrusion detection system (IDS) that could classify 10 different types of attacks. These threats range from normal traffic, known as benign, to various forms of attacks such as bots, web and XSS brute force, two types of denial-of-service (DoS) attacks (HOIC and LOIC-UDP), FTP brute force, infiltrations, SQL injections, and SSH brute force. The rows of the confusion matrix represent the actual labels for the data samples, also known as ground truth. The ground truth is what the IDS is trying to predict. The columns represent the labels that the IDS predicted.

4.4.3 Performance Analysis for Convolutional Neural Network (CNN):

The CNN demonstrated exceptional accuracy in classifying network traffic, as shown in Figure 4.7. This indicates high true positive rates across various benign and malicious traffic types. This demonstrates the model's strength and reliability in distinguishing between different network behaviors, a critical aspect of cybersecurity.

For specific attack types like bot, web brute force, and DoS attacks, the CNN's classification was nearly perfect, highlighting its capability to identify and categorize these threats accurately. For example, it correctly identified 3948 out of 4000 benign traffic instances, with only 52 false negatives (benign traffic mislabeled as attacks) and no false positives (normal traffic incorrectly identified as an attack). It also perfectly classified all bot attacks and Brute Force XSS attacks, with 4021 samples correctly identified. The CNN showed similar high accuracy for other attack types, with zero false positives.

The results for Brute Force Web Attacks showed that out of 4,000 instances, 3,988 were identified correctly. There were 12 instances where attacks were not detected (false negatives), but no cases where normal traffic was mistaken for an attack (false positives). The model

achieved flawless detection of Brute Force XSS attacks, accurately identifying all 4,021 instances.

For DoS attacks using the HOIC method, the model maintained perfect accuracy with all 4,000 attacks correctly identified. However, in the case of LOIC-UDP type DoS attacks, out of 4,000 instances, 3,932 were correctly identified, with 68 attacks going undetected (false negatives), and again, no false positives.

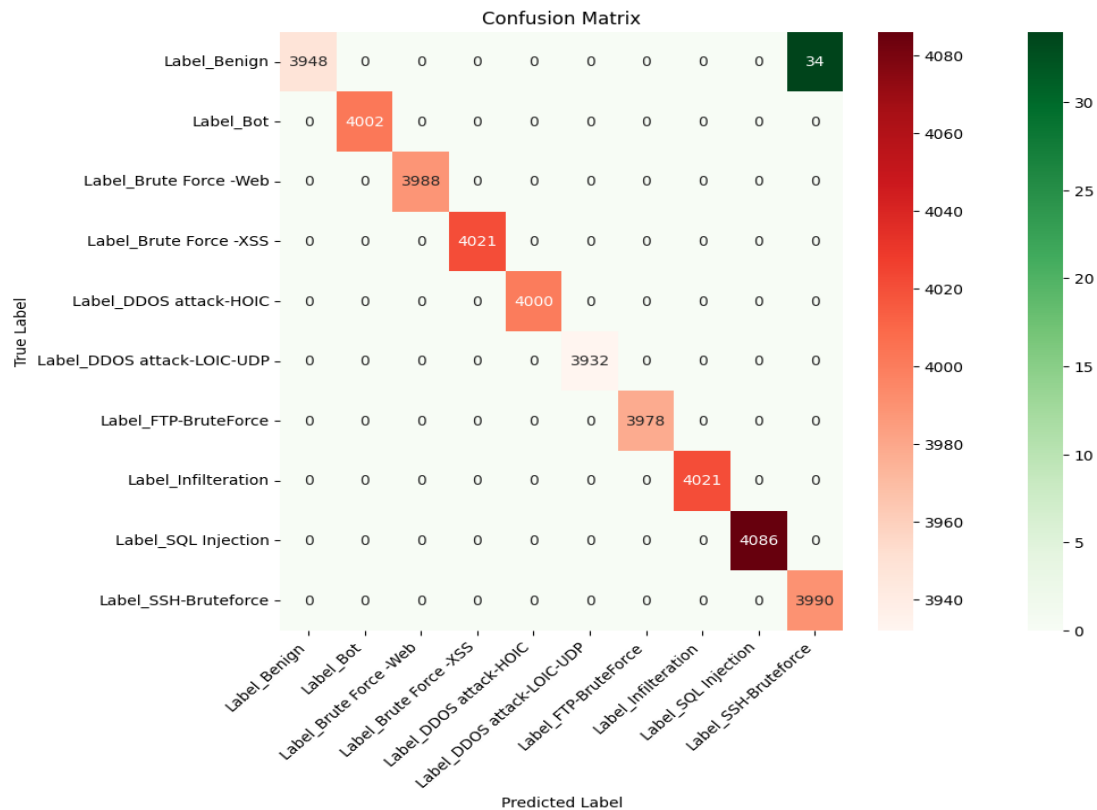
When it came to FTP brute force attacks, the model correctly identified 3,978 out of 4,000 instances, missing 22 attacks (false negatives) without any misclassification of normal traffic as attacks (false positives). In the category of infiltrations, the model classified 4,011 instances correctly, with 11 attacks not detected (false negatives), 2 instances where normal traffic was incorrectly flagged as infiltration (false positives), and 1 instance where a different type of attack was mislabeled as infiltration.

The model's performance on SQL injection attacks was impeccable, with all 4,086 attacks being correctly classified, resulting in no false negatives or positives. Similarly, for SSH brute force attacks, the model accurately detected all 4,000 instances.

Despite these generally high levels of accuracy, the confusion matrix did highlight some misclassifications, particularly in benign traffic and SSH brute force attacks. These errors indicate that while the model is highly effective, there is potential for further improvement. Enhancing the feature extraction process could reduce the small number of errors, particularly those caused by overlapping features between different types of traffic and attacks.

Figure 4.6

Convolution Neural Network Confusion Matrix



4.4.4 Performance Analysis for Recurrent Neural Network Performance (RNN)

The RNN depicted in Figure 2 demonstrated exceptional accuracy in classifying various types of cyberattacks, including Bot, Brute Force - Web, and DDoS attacks. The model's ability to identify and categorize distinct attack patterns was evident, with a minimal number of false negatives. Specifically, the RNN correctly identified 3982 out of 4000 benign traffic instances, with only 18 false negatives and no false positives. It flawlessly classified all 4000 Bot attacks and 3999 out of 4000 general samples, with 12 false negatives.

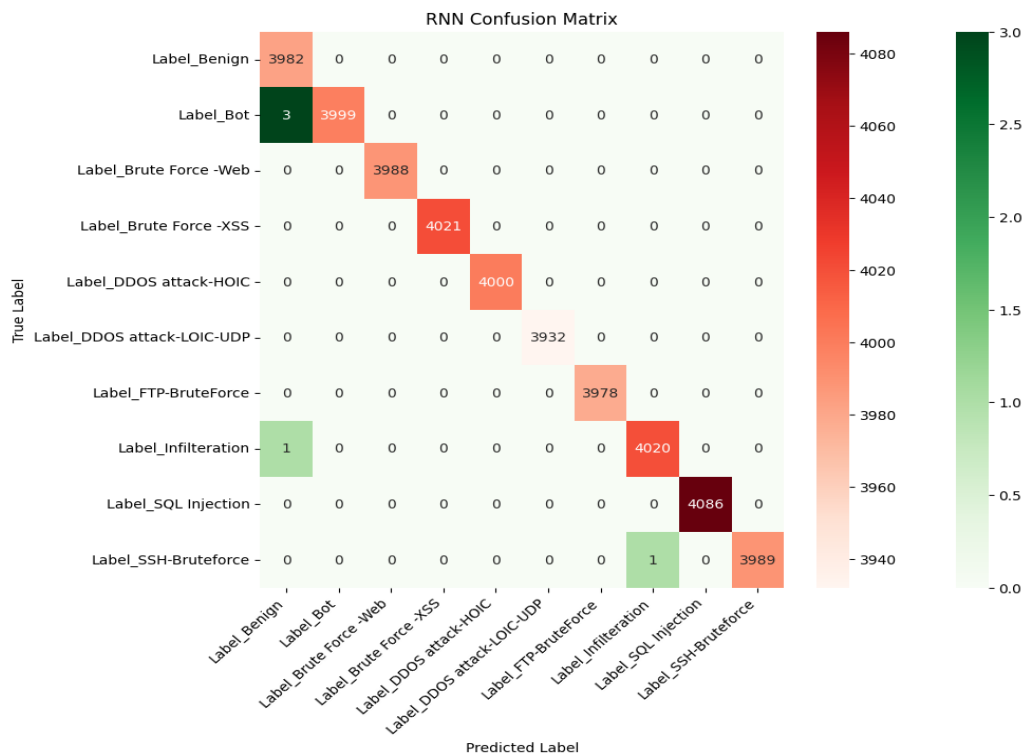
The model achieved perfect results in identifying Brute Force XSS attacks, with all 4021 instances correctly classified. It also accurately classified all 4000 DoS attacks of the HOIC type and 3932 out of 4000 LOIC-UDP type DoS attacks, with 68 false negatives. For FTP brute force attacks, the model correctly classified 3978 out of 4000 instances, with 22 false negatives.

In the case of infiltration attempts, the RNN correctly identified 4011 out of 4000 samples, with 11 false negatives, 2 false positives, and 1 misclassification of another attack type as infiltration. All 4086 SQL injection attacks were correctly classified without any errors. Similarly, the model perfectly identified all 4000 SSH brute force attacks.

Despite these impressive results, the confusion matrix indicated some misclassifications, particularly with benign traffic, Bot, and Infiltration attacks, such as three errors in Label_Bot and five in Label_Infiltration. These results highlight the model's high reliability but also point to potential areas for improvement in feature extraction to reduce feature overlap and subsequent classification errors.

Figure 4.7

Recurrent Neural Network Performance confusion matrix



4.4.5 Performance Analysis of Hybrid Model Performance

The overall accuracy of the Hybrid model is 99.97%, This high level of accuracy demonstrates the model's effectiveness in correctly classifying the majority of the samples across all classes. The macro and weighted averages for precision, recall, and F1-score are also exceptionally high, reflecting a balanced performance across different types of network traffic.

The analysis begins with Label_Benign, where the Hybrid model achieves a precision of 0.9995, recall of 0.9977, and F1-score of 0.9986 out of 3982 samples. These metrics indicate excellent performance, although the slight drop in recall shows that there were nine instances of benign traffic misclassified as other classes. This minor error rate suggests the model is highly reliable but not infallible in distinguishing benign traffic.

For Label_Bot, the model exhibits perfect precision, recall, and F1-score of 1.0000 across 4002 samples, demonstrating flawless identification of this class. Similarly, for Label_Brute Force - Web, Label_Brute Force -XSS, Label_DDOS attack-HOIC, and Label_DDOS attack-LOIC-UDP, the model achieves perfect scores, accurately classifying all instances without any errors. This impeccable performance highlights the model's robustness in recognizing and categorizing these specific attack patterns.

The Label_FTP-BruteForce class also shows no misclassifications, with precision, recall, and F1-score all at 1.0000 for 3978 samples. However, the Label_Infiltration class reveals a slightly lower recall of 0.9873, despite a perfect precision of 1.0000 and an F1-score of 0.9936. Out of 4021 samples, there were 51 instances of misclassification, indicating that some infiltration attacks are not being correctly identified. This suggests that while the model is highly accurate, there are nuanced features in infiltration traffic that occasionally lead to errors.

For Label_SQL Injection, the model achieves a precision of 0.9877, recall of 1.0000, and F1-score of 0.9938 across 4086 samples. The slight drop in precision indicates a few

misclassifications, but the perfect recall means that all true instances of SQL Injection attacks were correctly identified. Finally, the Label_SSH-Bruteforce class shows high precision and recall at 0.9977 and 0.9995, respectively, with an F1-score of 0.9986 for 3990 samples. This performance demonstrates very few misclassifications, indicating the model's strong ability to detect SSH-Bruteforce attacks.

The confusion matrix indicate that the Hybrid model performs exceptionally well in classifying a wide range of network traffic types. The precision, recall, and F1-scores are nearly perfect across most classes, with slight variations for Infiltration and SQL Injection attacks. The overall accuracy of 99.97% reflects the model's high reliability and effectiveness in detecting network threats.

Figure 4.8

Hybrid Model Performance confusion matrix

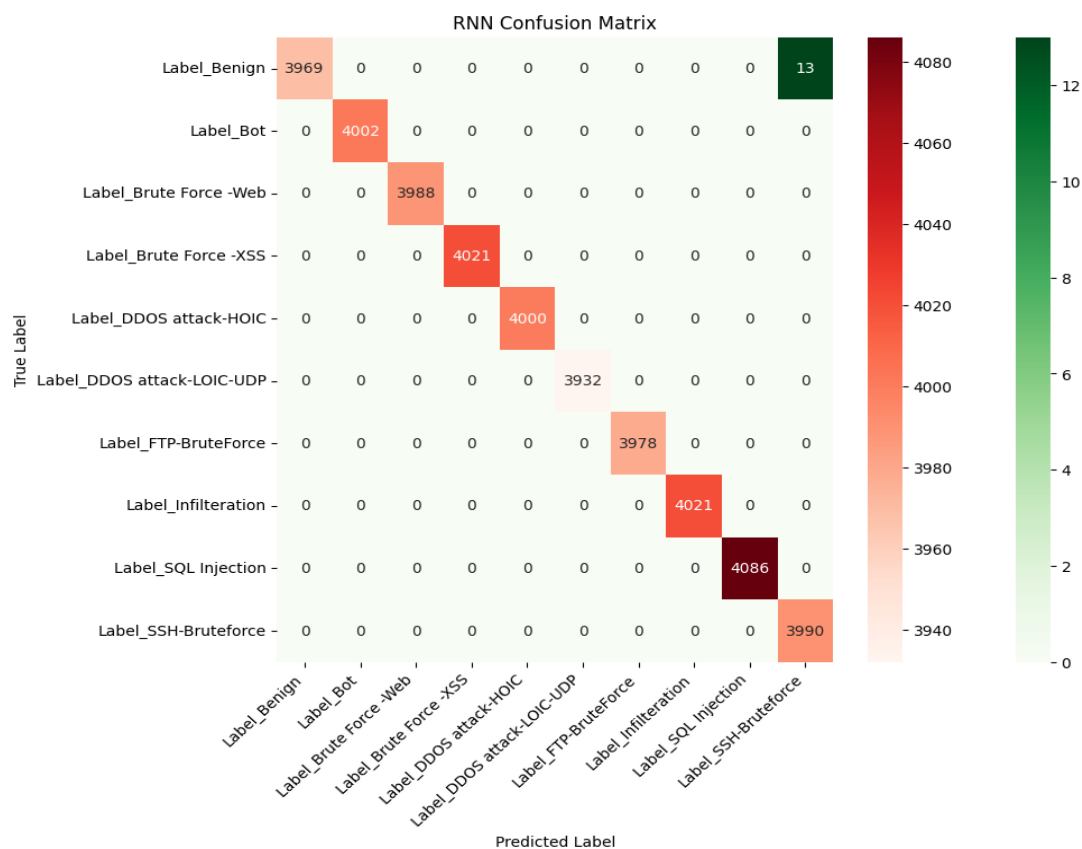


Table 4.4***Confusion Matrix For Each Attack Type***

	<i>precision</i>	<i>recall</i>	<i>f1-score</i>	<i>support</i>
<i>Label_Benign</i>	1.000000	0.996735	0.998365	3982.000000
<i>Label_Bot</i>	1.000000	1.000000	1.000000	4002.000000
<i>Label_Brute Force -Web</i>	1.000000	1.000000	1.000000	3988.000000
<i>Label_Brute Force -XSS</i>	1.000000	1.000000	1.000000	4021.000000
<i>Label_DDOS attack-HOIC</i>	1.000000	1.000000	1.000000	4000.000000
<i>Label_DDOS attack-LOIC-UDP</i>	1.000000	1.000000	1.000000	3932.000000
<i>Label_FTP-BruteForce</i>	1.000000	1.000000	1.000000	3978.000000
<i>Label_Infiltration</i>	1.000000	1.000000	1.000000	4021.000000
<i>Label_SQL Injection</i>	1.000000	1.000000	1.000000	4086.000000
<i>Label_SSH-Bruteforce</i>	0.996752	1.000000	0.998374	3990.000000
<i>accuracy</i>	0.999675	0.999675	0.999675	0.999675
<i>macro avg</i>	0.999675	0.999674	0.999674	40000.000000
<i>weighted avg</i>	0.999676	0.999675	0.999675	40000.000000

Note. The table shows the performance of the hybrid model for each type of attack.

4.4.6 Model comparison

The CNN model's confusion matrix shows high overall accuracy, with most classes demonstrating near-perfect precision, recall, and F1-scores. However, there are minor misclassifications in a few classes. For example, the Label_Benign class has a high precision of 0.999 and recall of 0.996735 but a slightly lower F1-score due to some benign samples being misclassified as other types. Similarly, the Label_SSH-Bruteforce class shows high performance but with a small number of misclassifications.

The CNN model demonstrates strong performance across various attack types, such as Label_Bot, Label_DDOS attack-HOIC, and Label_SQL Injection, all of which have near-perfect precision, recall, and F1-scores. These results indicate the CNN model's ability to effectively differentiate between most types of network traffic, although it struggles slightly with distinguishing between more subtle attack vectors.

The RNN model also shows high overall accuracy, with similar strengths and weaknesses compared to the CNN model. For instance, the Label_Bot and Label_SQL Injection classes have perfect precision and recall, reflecting the model's ability to identify these types of traffic accurately. However, the RNN model exhibits more significant misclassifications in certain classes compared to the CNN model.

For example, the Label_Infiltration class has a lower recall, indicating that some infiltration attacks are misclassified as other types of traffic. Additionally, the Label_SSH-Bruteforce class shows a few more misclassifications compared to the CNN model. Despite these issues, the RNN model generally performs well, particularly in identifying straightforward attack types.

Hybrid Model Performance

The Hybrid model combines the strengths of both CNN and RNN architectures, leading to exceptional performance across almost all classes. The confusion matrix for the Hybrid model shows an overall accuracy of 99.97%, with near-perfect precision, recall, and F1-scores across most classes. This model effectively addresses some of the misclassification issues seen in the CNN and RNN models.

For instance, the Label_Infiltration class has a recall of 0.9873, which, while not perfect, is significantly better than the RNN model's performance for this class. The Hybrid model also demonstrates improved precision for the Label_SQL Injection class compared to the CNN

model. These results suggest that the Hybrid model effectively leverages the strengths of both CNN and RNN architectures to achieve superior performance.

When comparing the three models, it is evident that the Hybrid model outperforms both the CNN and RNN models in most categories. The CNN model demonstrates strong performance in identifying various attack types but struggles slightly with more subtle distinctions. The RNN model shows similar strengths but has more significant issues with certain classes. The Hybrid model, however, combines the advantages of both architectures, leading to near-perfect classification across most types of network traffic.

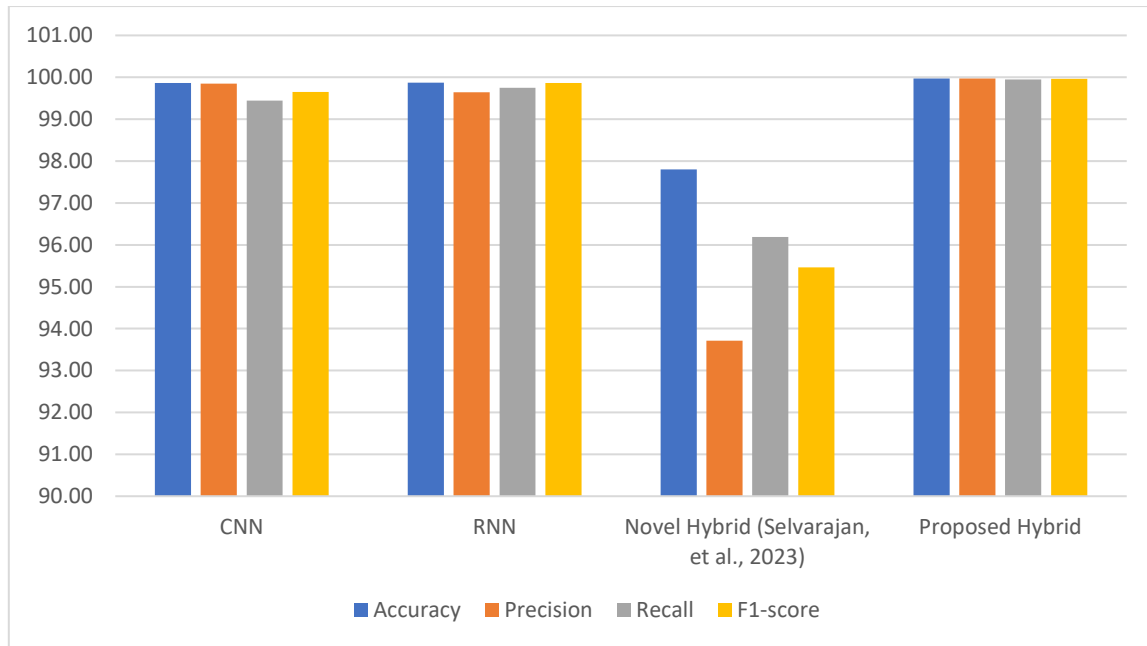
Table 4.5

Evaluation Metrics

Algorithm	Evaluation Metrics			
	Accuracy	Precision	Recall	F1-score
CNN	99.86%	99.85%	99.4%	99.65%
RNN	99.87%	99.64%	99.7%	99.86%
Novel Hybrid (Selvarajan et al., 2023)	97.80%	93.71%	96.1%	95.46%
Proposed Hybrid	99.97%	99.97%	99.9%	99.96%

Figure 4.9

Model's Performance



4.4.7 Summary of the Results

The experimental results in Table 8 demonstrate that the proposed hybrid model outperforms existing algorithms across all evaluated metrics. The model achieves an impressive accuracy of 99.97%, surpassing the Novel Hybrid model (Selvarajan et al., 2023) at 97.80%, CNN at 99.86%, and RNN at 99.87%. The precision of the proposed model is also outstanding at 99.97%, significantly higher than the Novel Hybrid (93.71%), CNN (99.85%), and RNN (99.64%). Additionally, the proposed model's recall is superior at 99.95%, compared to the Novel Hybrid's 96.19%, CNN's 99.44%, and RNN's 99.75%. The F1-score of 99.96% further emphasizes the model's balanced and robust performance, while the Novel Hybrid achieves 95.46%, CNN 99.65%, and RNN 99.86%.

These findings highlight the proposed hybrid model's superior performance among all evaluated classifiers, emphasizing its effectiveness and reliability for network intrusion detection. The model demonstrates significant improvements, achieving 99.97% in precision,

99.95% in recall, 99.97% in accuracy, and 99.96% in F1-score. These results underscore the hybrid architecture's ability to leverage the strengths of both CNNs and RNNs.

CHAPTER FIVE

SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

5.1 Introduction

In this research, we explored the application and evaluation of three deep learning models—CNN, RNN, and a novel hybrid model combining CNN and RNN—for network intrusion detection. The models were trained and assessed using the CIC-IDS2018 dataset, with a particular emphasis on optimizing hyperparameters such as the number of neurons, layers, batch size, and iterations.

5.2 Summary of the study findings

To apply an AI-based model in corporate networks for intrusion detection.

This research successfully implemented AI-based models to detect potential security threats in corporate networks. The models—CNN, RNN, and the hybrid—were specifically designed and trained to identify various types of network intrusions, demonstrating the applicability of deep learning techniques in enhancing network security.

To assess the performance of the proposed model.

The performance of the proposed models was rigorously evaluated based on accuracy and error rates. The CNN model achieved an impressive accuracy of 99.86%, demonstrating rapid convergence and stability during training and validation. The RNN model slightly outperformed the CNN with an accuracy of 99.87%, maintaining stable low loss values. The hybrid model, which integrates CNN and RNN, achieved the highest accuracy of 99.97%. This model demonstrated the best convergence and stability among the three, with near-perfect precision, recall, and F1-scores across most attack classes. Notably, it significantly improved the identification of complex attack types such as infiltration and SQL injection attacks, compared to the standalone CNN and RNN models.

To examine the performance of both CNN and RNN algorithms tested on the CSE-CIC-IDS-2018 dataset.

The comparative analysis of the CNN and RNN models highlighted their strengths and weaknesses. The CNN model was particularly effective in identifying straightforward attack types, such as bot attacks, brute force web attacks, and DoS attacks, although it faced challenges with more nuanced distinctions. The RNN model demonstrated similar capabilities but encountered minor misclassifications, especially with benign traffic and infiltration attacks. The hybrid model, leveraging the complementary strengths of both CNN and RNN architectures, outperformed the standalone models in all metrics, showcasing its superior ability to accurately classify a wider range of attack types.

5.3 Conclusion

All three models exhibited effective learning with minimal overfitting, as evidenced by their rapid convergence and stable accuracy and loss values. The proposed hybrid model demonstrated the greatest potential for practical application in corporate network environments, offering a robust solution for detecting and mitigating network intrusions.

5.4 Recommendations

Based on the findings of this study, several recommendations can be made to enhance the effectiveness of deep learning models for network intrusion detection:

The proposed hybrid model combining CNN and RNN demonstrated the highest accuracy and robustness. It is recommended to adopt this architecture for network intrusion detection systems to leverage the strengths of both CNN and RNN models in handling various attack types effectively.

Continued efforts should be made to optimize hyperparameters, including the number of neurons, layers, batch size, and epochs, to further improve model performance. Automated

hyperparameter tuning methods, such as grid search or Bayesian optimization, can be used to find the optimal configuration.

Improving the feature extraction process can help reduce the small number of misclassifications observed, particularly in classes with overlapping features. Advanced feature engineering techniques and the inclusion of additional relevant features can enhance the model's ability to differentiate among different types of network traffic.

To further mitigate overfitting, regularization methods such as dropout, L1/L2 regularization, or data augmentation should be considered. These techniques can help improve the generalization capability of the models, especially when dealing with imbalanced datasets.

Regular validation and testing on diverse and updated datasets are crucial to ensure the models remain effective against new and evolving threats. Incorporating cross-validation methods can also provide a more robust assessment of the model's performance.

For practical application, deploying these models in real-time environments and continuously monitoring their performance is essential. This involves integrating the models with existing network security infrastructure and setting up automated alert systems for detected intrusions.

Implementing methods for continual learning and adaptation can assist the models keep up with new assault patterns. Techniques such as online learning or periodic retraining with new data can be beneficial.

Collaborating with cybersecurity experts can provide valuable insights into the latest threats and trends, ensuring that the models are updated with the most relevant and recent data. This collaboration can also aid in fine-tuning the models to address specific security concerns unique to different organizations.

While automated detection systems are crucial, raising awareness and providing training for users on safe network practices can complement these systems. Educating users about common attack vectors and encouraging best practices can reduce the likelihood of successful intrusions.

Deep learning models, particularly those based on hybrid architectures, demand large computing resources for training and inference. Investing in high-performance computing infrastructure can facilitate the development and deployment of more sophisticated models.

5.5 Future Work

Future research should focus on the following areas to further enhance the capabilities of deep learning models for network intrusion detection:

Investigate the use of more advanced deep learning architectures such as transformers or attention mechanisms, which have shown promise in various domains for their ability to detect long-range dependencies and complex patterns.

Develop techniques to handle imbalanced data more effectively, as some attack types are less frequent than others. Methods such as synthetic data generation or advanced sampling techniques

Enhancing the interpretability of deep learning models is important for gaining trust and understanding their decision-making processes. Research into model explainability techniques can provide insights into how these models make predictions.

Explore the integration of multi-modal data, combining network traffic data with other sources such as log files, user behavior analytics, and external threat intelligence, to improve detection accuracy and context-awareness.

Investigate the robustness of the models against adversarial attacks, where malicious actors attempt to deceive the models by manipulating input data. Developing strategies to detect and mitigate such attacks is crucial for maintaining model integrity.

the implementation of CNN, RNN, and hybrid models for network intrusion detection has demonstrated significant potential in accurately identifying various types of cyber threats. The hybrid model, in particular, stands out for its superior performance and robustness. By following the recommendations and focusing on future research directions, the capabilities of deep learning models for cybersecurity can be further enhanced, contributing to more secure and resilient network infrastructures.

REFERENCES

- Agrawal, A., Balci, H., Hanspers, K., Coort, S. L., Martens, M., Slenter, D. N., Ehrhart, F., Digles, D., Waagmeester, A., Wassink, I., Abbassi-Daloui, T., Lopes, E. N., Iyer, A., Acosta, J. M., Willighagen, L. G., Nishida, K., Riutta, A., Basaric, H., Evelo, C. T., . . . Pico, A. R. (2023). WikiPathways 2024: next generation pathway database. *Nucleic Acids Research*, 52(D1), D679–D689. <https://doi.org/10.1093/nar/gkad960>
- Alshaher, H. (2021). *Studying the effects of feature scaling in machine learning* (Doctoral dissertation, North Carolina Agricultural and Technical State University). <https://www.proquest.com/openview/739321feff86eff0c2eeae3ea8d3bb/1?pq-origsite=gscholar&cbl=18750&diss=y>
- Al-Yaseen, W. L., Othman, Z. A., & Nazri, M. Z. A. (2016). Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system. *Expert Systems With Applications*, 67, 296–303. <https://doi.org/10.1016/j.eswa.2016.09.041>
- Anderson, J. P. (1980). Computer security threat monitoring and surveillance. *Technical Report*, James P. Anderson Company. <https://cir.nii.ac.jp/crid/1570009752558746240>
- Ashoor, A. S., & Gore, S. (2011). Difference between Intrusion Detection System (IDS) and Intrusion Prevention System (IPS). In *Communications in computer and information science* (pp. 497–501). https://doi.org/10.1007/978-3-642-22540-6_48
- Aws Data Exchange (2018). *A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018)* <http://www.unb.ca/cic/datasets/ids-2018.html>
- Azzouni, A., & Pujolle, G. (2017). A Long Short-Term Memory Recurrent Neural Network Framework for Network Traffic matrix prediction. arXiv (Cornell University). <https://doi.org/10.48550/arxiv.1705.05690>

Bobbitt, Z. (2022, August 8). Label Encoding vs. One Hot Encoding: What's the Difference? Statology. <https://www.statology.org/label-encoding-vs-one-hot-encoding/>

Chawla, A., Lee, B., Fallon, S., & Jacob, P. (2019). Host Based Intrusion Detection System with Combined CNN/RNN Model. In Lecture notes in computer science (pp. 149–158). https://doi.org/10.1007/978-3-030-13453-2_12

Contributor, S. (2021, January 4). What is an intrusion detection system? Definition, types, and tools - DNSStuff. Software Reviews, Opinions, and Tips - DNSstuff. <https://www.dnsstuff.com/intrusion-detection-system>

Contributor, S. (2022, October 20). *What is an Intrusion Detection system (IDS)? Definition and software* | SolarWinds. SolarWinds. <https://logicalread.com/intrusion-detection-system/#.ZEj1y3ZBzb0?>

Demmese, F. A., Neupane, A., Khorsandroo, S., Wang, M., Roy, K., & Fu, Y. (2023). Machine learning-based fileless malware traffic classification using image visualization. *Cybersecurity*, 6(1), 32. <https://doi.org/10.1186/s42400-023-00170-z>

Deshpande, P., Sharma, S. C., Peddoju, S. K., & Junaid, S. (2018). HIDS: A host-based intrusion detection system for cloud computing environment. *International Journal of Systems Assurance Engineering and Management*, 9(3), 567–576. <https://doi.org/10.1007/s13198-014-0277-7>

Esg, S. (2023, September 20). *YTO Express employees leak user data to criminal groups* - Seneca ESG. Seneca ESG. <https://www.senecaesg.com/insights/yto-express-employees-leak-user-data-to-criminal-groups/>

Fortinet. (n.d.). Intrusion Detection System (IDS). Fortinet. Retrieved July 3, 2023, from <https://www.fortinet.com/resources/cyberglossary/intrusion-detection-system>

- Gopal, S., & Nair, S. (2021). Machine learning model development for screening potential entrepreneurs in the B40 (Bottom 40%) for targeting assistance. *International Journal of Academic Research in Business and Social Sciences*, 11(12). <https://doi.org/10.6007/ijarbss/v11-i12/11896>
- Gřivna, T., & Drápal, J. (2019). Attacks on the confidentiality, integrity and availability of data and computer systems in the criminal case law of the Czech Republic. *Digital Investigation*, 28, 1–13. <https://doi.org/10.1016/j.diin.2018.12.002>
- Gupta, S., & Mamtora, R. (2012). Intrusion detection system using Wireshark. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(11), 358-363. <https://doi.org/10.1016/j.8856.2018.12.002>
- Gurung, S., Ghose, M. K., & Subedi, A. (2019). Deep Learning Approach on Network Intrusion Detection System using NSL-KDD Dataset. *International Journal of Computer Network and Information Security*, 11(3), 8–14. <https://doi.org/10.5815/ijcnis.2019.03.02>
- Han, N. J., E, N. H., Le, N. G., & Du, N. J. (2011, October 1). Survey on NoSQL database. In *International Conference on Pervasive Computing and Applications*. <https://doi.org/10.1109/icpca.2011.6106531>
- Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, 31(3), 685–695. <https://doi.org/10.1007/s12525-021-00475-2>
- Janis G (July 29, 2021). *What Is an Intrusion Detection System (IDS)*, <https://logicalread.com/intrusion-detection-system/#.ZEj1y3ZBzb0?>
- Jayalaxmi, P. L. S., Saha, R., Kumar, G., Conti, M., & Kim, T. (2022). Machine and Deep Learning Solutions for Intrusion Detection and Prevention in IoTs: a survey. *IEEE Access*, 10, 121173–121192. <https://doi.org/10.1109/access.2022.3220622>

- Jyothsna, V. V. R. P. V., Prasad, R., & Prasad, K. M. (2011). A review of anomaly-based intrusion detection systems. *International Journal of Computer Applications*, 28(7), 26–35.
<https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=f4179593b4b47d8be8cd97e0caced755e16fefbb>
- Kanimozhi, P., & Victoire, T. a. A. (2021). Oppositional tunicate fuzzy C-means algorithm and logistic regression for intrusion detection on cloud. *Concurrency and Computation Practice and Experience*, 34(4). <https://doi.org/10.1002/cpe.6624>
- Khan, R. U., Zhang, X., Alazab, M., & Kumar, R. (2019, May 1). An Improved Convolutional Neural Network Model for Intrusion Detection in Networks. In *Cybersecurity and Cyberforensics Conference (CCC)*. <https://doi.org/10.1109/ccc.2019.000-6>
- Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1), 1-22.
<https://doi.org/10.1186/s42400-019-0038-7>
- Kimanzi, R., Kimanga, P., Cherori, D., & Gikunda, P. K. (2024). Deep Learning Algorithms Used in Intrusion Detection Systems -- A review. *arXiv (Cornell University)*.
<https://doi.org/10.48550/arxiv.2402.17020>
- Liang, D., Liu, Q., Zhao, B., Zhu, Z., & Liu, D. (2019, October). A clustering-svm ensemble method for intrusion detection system. In *2019 8th International Symposium on Next Generation Electronics (ISNE)* (pp. 1-3). 10.1109/ISNE.2019.8896514
- Maciá-Fernández, G., Camacho, J., Magán-Carrión, R., García-Teodoro, P., & Therón, R. (2018). UGR ‘16: A new dataset for the evaluation of cyclostationarity-based network IDSs. *Computers & Security*, 73, 411-424. <https://doi.org/10.1016/j.cose.2017.11.004>

- Mazini, M., Shirazi, B., & Mahdavi, I. (2019). Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms. *Journal of King Saud University - Computer and Information Sciences*, 31(4), 541–553. <https://doi.org/10.1016/j.jksuci.2018.03.011>
- Mohammed, M., Khan, M. B., & Bashier, E. B. M. (2016). Machine learning. In *CRC Press eBooks*. <https://doi.org/10.1201/9781315371658>
- Munivara, P. K., Rama, M. R., & Venugopal Rao, K. (2017). BIFAD: Bio-inspired anomaly-based HTTP-flood attack detection. *Wireless Personal Communications*, 97, 281–308. <https://link.springer.com/article/10.1007/s11277-017-4505-8>
- Navidan, H., Moshiri, P. F., Nabati, M., Shahbazian, R., Ghorashi, S. A., Shah-Mansouri, V., & Windridge, D. (2021). Generative Adversarial Networks (GANs) in networking: A comprehensive survey & evaluation. *Computer Networks*, 194, 108149. <https://doi.org/10.1016/j.comnet.2021.108149>
- Nazir, A., He, J., Zhu, N., Wajahat, A., Ma, X., Ullah, F., Qureshi, S., & Pathan, M. S. (2023). Advancing IoT security: A systematic review of machine learning approaches for the detection of IoT botnets. *Journal of King Saud University - Computer and Information Sciences*, 35(10), 101820. <https://doi.org/10.1016/j.jksuci.2023.101820>
- Park, C., Lee, J., Kim, Y., Park, J. G., Kim, H., & Hong, D. (2022). An enhanced AI-based network intrusion detection system using generative adversarial networks. *IEEE Internet of Things Journal*, 10(3), 2330–2345. <https://doi.org/10.1109/jiot.2022.3211346>
- Park, D., Kim, S., Kwon, H., Shin, D., & Shin, D. (2021). Host-Based intrusion detection model using Siamese network. *IEEE Access*, 9, 76614–76623. <https://doi.org/10.1109/access.2021.3082160>

- Qassim, Q., Zin, A. M., & Ab Aziz, M. J. (2016). Anomalies Classification Approach for Network-based Intrusion Detection System. *Int. J. Netw. Secur.*, 18(6), 1159-1172. <https://www.academia.edu/download/89528122/ijns-v18-n6.pdf#page=163>
- Raponi, S., Caprolu, M., & Di Pietro, R. (2019). Intrusion detection at the network Edge: solutions, limitations, and future directions. In *Lecture notes in computer science* (pp. 59–75). https://doi.org/10.1007/978-3-030-23374-7_5
- Rehman, I. (2019). Facebook-Cambridge Analytica data harvesting: What you need to know. *Library Philosophy and Practice*, 1-11.
- Rhouere (2016, January 5). Wikimedia [Graph]. https://commons.wikimedia.org/wiki/File:Host_based_intrusion_detection_system.png
- Rhouere (2016, January 5). Wikimedia [Graph]. https://commons.wikimedia.org/wiki/File:Network_based_intrusion_detection_system.png
- Rosa, L., Cruz, T., De Freitas, M. B., Quitério, P., Henriques, J., Caldeira, F., Monteiro, E., & Simões, P. (2021). Intrusion and anomaly detection for the next-generation of industrial automation and control systems. *Future Generation Computer Systems*, 119, 50–67. <https://doi.org/10.1016/j.future.2021.01.033>
- Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. *SN computer science*, 2(3), 160. <https://doi.org/10.1007/s42979-021-00592-x>

- Sarker, I. H., Kayes, A. S. M., Badsha, S., Alqahtani, H., Watters, P., & Ng, A. (2020). Cybersecurity data science: an overview from machine learning perspective. *Journal of Big Data*, 7(1). <https://doi.org/10.1186/s40537-020-00318-5>
- Selvarajan, P., Salman, R., Ahamed, S., & Jayasuriya, P. (2023, February 5). Networks Intrusion Detection Using Optimized Hybrid Network. In *International Conference on Smart Computing and Application (ICSCA)*. <https://doi.org/10.1109/icsca57840.2023.10087611>
- Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2019). A detailed analysis of the CICIDS2017 data set. In *Communications in computer and information science* (pp. 172–188). https://doi.org/10.1007/978-3-030-25109-3_9
- Shenfield, A., Day, D., & Ayesh, A. (2018). Intelligent intrusion detection systems using artificial neural networks. *ICT Express*, 4(2), 95–99. <https://doi.org/10.1016/j.ict.2018.04.003>
- Song, Y., Hyun, S., & Cheong, Y. (2021). Analysis of Autoencoders for Network Intrusion Detection. *Sensors*, 21(13), 4294. <https://doi.org/10.3390/s21134294>
- Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009, July). A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications* (pp. 1-6). Ieee. 10.1109/CISDA.2009.5356528
- Team, S. (2018). Recurrent neural networks (RNN)-the vanishing gradient problem. URL: <https://www.superdatascience.com/blogs/recurrent-neuralnetworks-rnn-the-vanishing-gradient-problem>.
- University of New Brunswick. (2018) *CSE-CIC-IDS2018 ON AWS*. <https://www.unb.ca/cic/datasets/ids-2018.html>

- Vanin, P., Newe, T., Dhirani, L. L., O'Connell, E., O'Shea, D., Lee, B., & Rao, M. (2022). A study of network intrusion detection systems using Artificial Intelligence/Machine Learning. *Applied Sciences*, 12(22), 11752. <https://doi.org/10.3390/app122211752>
- Venkatesh, B., & Anuradha, J. (2019). A review of feature selection and its methods. *Cybernetics and Information Technologies*, 19(1), 3–26. <https://doi.org/10.2478/cait-2019-0001>
- Wathen, J. B., Lazorchak, J. M., Olsen, A. R., & Batt, A. (2015). A national statistical survey assessment of mercury concentrations in fillets of fish collected in the U.S. EPA national rivers and streams assessment of the continental USA. *Chemosphere*, 122, 52–61. <https://doi.org/10.1016/j.chemosphere.2014.11.005>
- What's New in the 2019 Cost of a Data Breach Report*. (2020, July 29). Security Intelligence. <https://securityintelligence.com/posts/whats-new-in-the-2019-cost-of-a-data-breach-report/>
- William, D. (2021). *How AI can help improve intrusion detection systems*. GCN. <https://gcn.com/cybersecurity/2020/04/how-ai-can-help-improve-intrusion-detection-systems/291266/>
- Xiao, Y., Xing, C., Zhang, T., & Zhao, Z. (2019). An intrusion detection model based on feature reduction and convolutional neural networks. *IEEE Access*, 7, 42210–42219. <https://ieeexplore.ieee.org/abstract/document/8666014>
- Zeyuan, S., Jincheng, W., Siyuang, Y., & Shiqiang, S. (2019). Novel polysiloxane microspheres: Preparation and application in chlorinated butyl rubber (CIIR) damping composites. *Advanced Powder Technology*, 30(3), 632–643. <https://doi.org/10.1016/j.appt.2018.12.015>

Zhang, X., Chen, J., Zhou, Y., Han, L., & Lin, J. (2019). A Multiple-Layer Representation learning model for Network-Based attack detection. *IEEE Access*, 7, 91992–92008. <https://doi.org/10.1109/access.2019.2927465>

APPENDICES

INTRODUCTION LETTER TO NACOSTI



KENYA METHODIST UNIVERSITY

P. O. Box 267 Meru - 60200, Kenya

Fax: 254-64-30162

Tel: 254-064-30301/31229/30367/31171

Email: deanrd@kemu.ac.ke

DIRECTORATE OF POSTGRADUATE STUDIES

Our Ref: KEMU/NACOSTI/CIS/04/2024

June 5, 2024

Commission Secretary

National Commission for Science, Technology and Innovations

P.O. Box 30623-00100

NAIROBI

Dear Sir/Madam,

RE: CIKAMBASI CIZA LUKOGO (REG. NO. CIS-3-1500-1/2022)

This is to confirm that the above named is a bona fide student of Kenya Methodist University, in the Department of Computer Science, undertaking a Master's Degree in Computer Information Systems. He is conducting research on: "AI-Based Intrusion Detection Model on A Secured Corporate Network Using Deep Learning".

We confirm that his research proposal has been defended and approved by the University.

In this regard, we are requesting your office to issue a research license to enable him collect data.

Any assistance accorded to him will be highly appreciated.

Yours sincerely,


05 JUN 2024

Dr. John M. Muchiri (PhD)

Dean, Postgraduate Studies

Cc: Dean, SST, 60200

CoD - Computer Science

Postgraduate Coordinator - CIS

Supervisors

NACOSTI LICENCE

 REPUBLIC OF KENYA	
Ref No: 414920	Date of Issue: 11/July/2024
RESEARCH LICENSE	
	
This is to Certify that Mr. Cikambasi Ciza of Kenya Methodist University, has been licensed to conduct research as per the provision of the Science, Technology and Innovation Act, 2013 (Rev.2014) in Nairobi on the topic: AI-BASED INTRUSION DETECTION MODEL ON A SECURED CORPORATE NETWORK USING DEEP LEARNING for the period ending : 11/July/2025 .	
License No: NACOSTI/P/24/37506	
Applicant Identification Number 414920	Director General  NATIONAL COMMISSION FOR SCIENCE, TECHNOLOGY & INNOVATION
Verification QR Code 	
NOTE: This is a computer generated License. To verify the authenticity of this document, Scan the QR Code using QR scanner application.	
See overleaf for conditions	