UNIVERSITY OF LEICESTER

School of Computing and Mathematical Sciences

CO7201 Individual Project

Final Report
ENHANCED INTRUSION DETECTION SYSTEM USING DEEP LEARNING

[Syed Faiz Ur Rahman]

[sfur1@student.le.ac.uk]

[249049717]

Project Supervisor: [Dr Babajide Afeni]
Principal Marker: [Dr Victoria Wright]

Word Count: 12055

Abstract

The proposed project strives to refine “intrusion detection systems” (IDS) with the help of “deep learning” (DL) algorithms to increase the identification of cyberattacks in IoT networks. The project is aimed at developing an innovative IDS that can use deep learning models, containing “Convolutional Neural Networks” (CNN), “Long Short-Term Memory” (LSTM) networks, and “Recurrent Neural Networks” (RNN) to precisely predict and categorise numerous network intrusions. The main goal is to improve the correctness and efficiency of predicting malicious network traffic with reference to DL algorithms that have the ability to learn to recognise complex patterns and time dependencies in IoT traffic data.

The project consists of gathering and preprocessing of “IoT network traffic data” and applying DL models to classify and predict the type of intrusions. The performance of the models is measured through ‘accuracy’, ‘precision’, ‘recall’, and ‘F1 score’ evaluation metrics. The efficiency in models is also evaluated by confusion matrices and the AUC-ROC curves. The results of the study prove that the RNN model is better than the CNN and the LSTM models, obtaining an accuracy rate of 98.55%. RNN is also demonstrating a high efficiency in identifying malicious activity in complex and exploratory attacks, like a DDoS, DoS, and Mirai Botnet attack.

The study fills in significant research gaps in the current IDS studies based on the need to track rare or complex attacks, as well as the inability of the existing models to work at a much larger scale. By incorporating a hybrid deep neural network architecture and confronting challenges such as the presence of unbalanced classes, the project will become a part of improving a more effective, resilient and scalable IDS in the context of IoT networks that potentially can be applied to real-time dynamics and enhanced with respect to resisting emerging threats.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Background

As a cybersecurity strategy, “Intrusion Detection System” (IDS) is widely employed to detect unauthorised entrance or cyberattacks carried out in an intellectual system or network traffic. IDS collects information and can monitor what is happening on the network and all over the system, in search of malicious activities or breaches of security policy. It can trace the hostile applications, like when hacked into something, malware and the development of gaps in the networking data (Mohale and Obagbuwa, 2025). One of the types of security systems includes a deep learning-based IDS that uses deep learning algorithms to identify and analyse anomalous behaviour in networks.

The neural networks, including the convoluted neural network or the recurrent neural network, have the ability to learn complex features of network traffic to detect malicious activities within huge amounts of cybersecurity data (Talukder et al., 2024). Neural networks have been used to define ordinary and unusual activities under the IDS to ensure that security solutions are available on time, and therefore, the cyber-attacks and unauthorised access can be considered as preventative actions.

Combining the traditional machine-based IDS and sophisticated AI model to create an "Enhanced Intrusion Detection System" (EIDS) allows for enhanced detection accuracy and reduces false positives in signifying attacks (Faker and Dogdu, 2019). EIDS has the benefit of recognising intricate and dynamic cyberattacks as well as insidious intrusions that machine learning algorithms cannot predict well (Chua and Salam, 2023). It incorporates neural network architectures to strengthen the efficiency of the system to analyse large amounts of data within a limited amount of time.

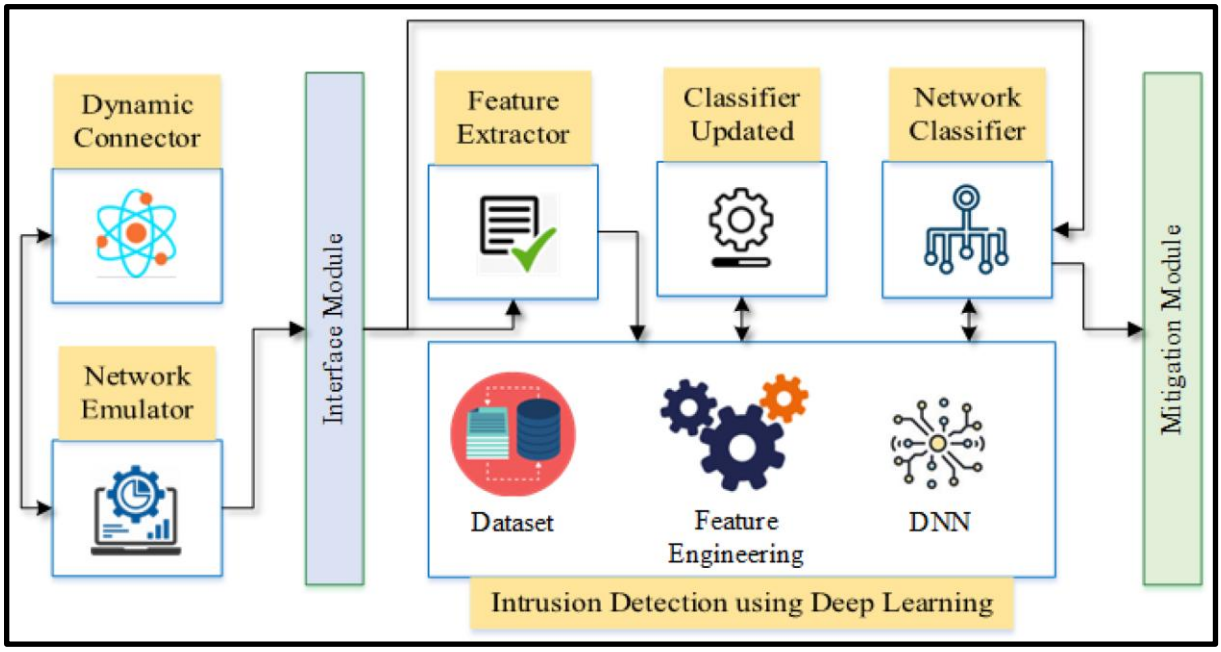


Figure 1.1: Application of deep learning for improved IDS

(Source: Awajan, 2023)

Figure 1.1 represents an emerging IDS based on deep learning. It consists of modules of dynamic connection, feature extraction, dataset processing, feature engineering, classifier update, network classification, and mitigation that help to identify and stop the network intrusion. Improved IDS using deep learning technology is impactful in proactive detection of new threats in reducing the probability of cyber theft, weighing better response mechanisms, and safeguarding personal data on a real-time basis (Al-Imran and Ripon, 2021).

1.2 Problem Statement

The intricacy of the networks of the "Internet of Things" (IoT) devices has increased, and this has also led to the susceptibility of several security threats owing to information breaches and cyberattacks. The existing IDS tend to be inefficient in detecting and identifying complex and dynamic attacks based on networks. Such systems are restricted only to highly advanced technologies, and they have not experienced diverse attacks randomly; hence, the accuracy and efficiency of their detection is very poor (Islam et al., 2024). Furthermore, the IoT networks have large network traffic, and it is tough to manually monitor and discover them. In the proposed project, the models of deep learning algorithms will be deployed to cover these issues. The solution can strengthen the accuracy, efficiency and trustworthiness of recognising malicious activities on IoT networks. The project will present elevated sensitivity of "malicious network traffic" with the use of the advanced methodologies of "Artificial

Intelligence". It will also create an intuitive user interface to deal with real-time interaction and management of alerts to achieve a comprehensive solution for security in IoT networks.

The traditional machine learning (ML) techniques have difficulties in processing high-dimensional and intricate data, as well as identifying new, subtle attacks in IDS problems (Hnamte et al., 2023). These processes require proper "feature engineering" steps as these lack appropriate and modern technologies in estimating inconsistent and evolving cyberthreats. These constraints are overcome through deep learning, where hierarchical representations can be learned with deep learning and it can automatically identify valuable features to learn (Vishwakarma and Kesswani, 2022). It improves IDS in terms of detecting sophisticated unknown attacks with better performance that is highly accurate and does not require individual feature selection, which makes it crucial in high-end cybersecurity applications.

1.3 Aim and Objectives

Aim

The aim of the project is to boost intrusion detection systems by predicting the label of the network intrusions with deep learning models, with efficient detection accuracy and efficiency.

Objectives

- ➢ To collect an Internet of Things Networks dataset of intrusions to detect and classify various network-based attacks with the help of the Python programming language.
- ➢ To implement deep learning models for intrusion detection in the IoT network traffic by employing "Convolutional Neural Networks" (CNNs), "Long Short-Term Memory (LSTM) networks", and "Recurrent Neural Networks" (RNNs).
- ➢ To identify the reliability as well as the accuracy of intrusion detection systems by applying advanced deep learning methodologies.
- ➢ To evaluate and compare the performance of deep learning algorithms in recognising malicious network traffic with the help of precision, recall, and F1 scores.
- ➢ To develop an intuitive interface of the IDS in such a way that a user can interact with and manage the real-time alerts on IoT network intrusions.

1.4 Research Questions

- ★ How can an IoT network dataset be properly collected and pre-processed for intrusion detection systems?

- ★ What deep learning algorithms achieve the most satisfactory accuracy for classifying network-based attacks?
- ★ How can deep learning models handle the constraints of traditional IDS in indicating advanced cyberattacks with improved accuracy and reliability?
- ★ What performance metrics can be used to evaluate the implementation of deep learning models in estimating IoT intrusions?
- ★ How can a user-friendly interface be developed to observe real-time IoT intrusion alerts appropriately?

1.5 Project Motivation

The project dwells on enhancing network system "intrusion detection" using DL models. Conventional methods of intrusion detection are mostly inadequate in performance evaluation due to the high number of cyber-attacks (Khan et al., 2022). Deep learning models promise to increase accuracy in detection due to being able to detect complex patterns and anomalies, which would be otherwise hidden in classical methods (Durgaraju and Vel, 2025). Through this project, an IDS is going to be developed that is able to classify known as well as unknown attacks, respond more quickly and reduce cyberattacks. It encompasses the integration of several DL models to analyse both network traffic data in a time-efficient and scalable form. The aim is to present a robust solution to reinforce security on the critical infrastructure, minimise the vulnerabilities and improve the overall trustworthiness of network protection systems (Maseno et al., 2024). Lastly, it is anticipated that the project will deliver more potent cybersecurity systems that would safeguard sensitive information and other important facilities against malicious attacks.

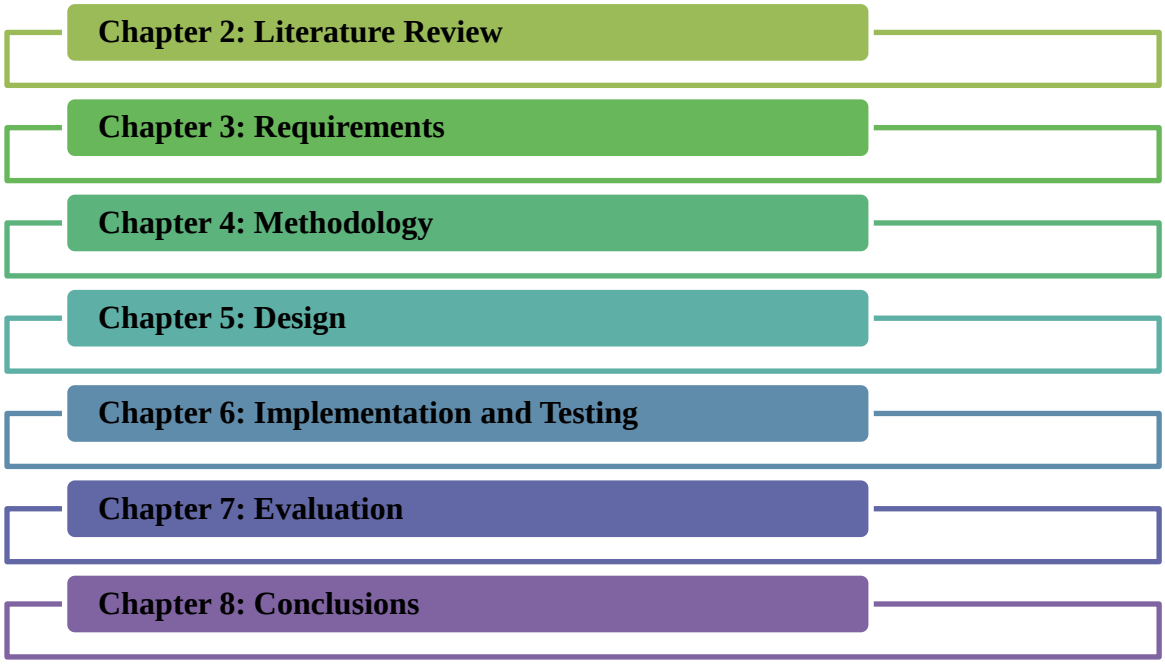
1.6 Project Rationale

Due to increasing complexity and volume of cyberattacks on the Internet of Things, more sophisticated intrusion-detection systems are in demand. The major factors that make IoT networks vulnerable are their massive size and complex data traffic, and these features attract cybercriminals. It is possible to address the challenges with neural network algorithms, which are efficient in terms of gaining more detection accuracy than traditional approaches (Mirsky et al., 2018). The rationale of conducting this work is based on the fact that there are no sufficient existing investigations about the applications of deep learning in the early detection of cyber threats in IoT networks.

The enhanced IDS analysis with deep learning architecture is not only more accurate but effective too, and adaptable in differentiating and forecasting benign attacks and malignant

attacks or aggressive attacks in a particular network traffic (Mohale and Obagbuwa, 2025). Being a part of the critical infrastructure of the digital world, IoT-based networks necessitate an efficient and easily scalable intrusion detection system. Besides, the establishment of a user-friendly interface ensures real-time tracking and a short response towards any potential security risks. The specified approach will help not only improve the degree of cyberprotection but also streamline the security of IoT networks.

1.7 Dissertation Structure



Chapter 2: Literature Review

2.1 Introduction

The chapter gives an extensive review of “Intrusion Detection Systems” (IDS) in IoT networks in relation to deep learning methods to enhance the efficacy of detection. It considers the multiple performance assessment systems, such as ‘precision’, ‘recall’, and ‘F1 score’, to evaluate the IDS performance. In addition, a theoretical framework is given to support the objectives of the study. The concern of using neural network algorithms and the comparison of different DL algorithms in the detection of network intrusion has also been covered in the chapter. It closes by signifying the gaps in the currently available literature and justification of the proposed project.

2.2 Overview of Intrusion Detection Systems (IDS) in IoT Networks

As reported by Isong, Kgote and Abu-Mahfouz, (2024), an IDS is a major point in the network of an organisation. They are utilised in seeking and interfering with those malignant acts, which in this case, would be efforts to gain unauthorised accessibility or malicious attacks on the network resources. An IDS is able to analyse data packets and system logs and therefore detect the various types of attacks, such as attempts of illicit parties to gain access, malware and “denial of service” (DoS) attacks. An illustrative example is that an IDS deployed on a corporate network could detect any abnormal traffic within an organisation that could be a “Distributed Denial of Service” (DDoS) attack and inform the administrators to take precautionary measures (Jablaoui and Liouane, 2025). An organisation can promptly act through an IDS in case of a possible malicious attack on the security of its information and systems, and hence retain the integrity and privacy of the organisation.

As stated by Elrawy, Awad and Hamed, (2018), the affiliation of various internet accessible devices places an IDS at the centre of the prevention and minimisation of security problems in the IoT networks. IDS, especially, will be of critical importance in an environment where there is so much density in terms of IoT device counts, all interrelated and varying capabilities. These systems can track activities on the network and device behaviours in an effort to identify abnormal patterns or fundamental behaviours and hence contribute towards the general security of the IoT ecosystems. Implementation of IDS on the IoT networks needs to keep in view the specificities of the IoT devices, such as poor processing capacity and memory, absence of energy resources (Shaker, Al-Musawi and Hassan, 2023).

However, classic methods in IDS might not be straightforward to use. These challenges can be mitigated through different methods, such as the use of lightweight anomaly detection

algorithms and incorporating deep learning methods to enhance high accuracy in the detection without affecting the performance (Bamber et al., 2025). An example is that Suricata was used as a network cyberthreat recognition system in smart homes, and it provides a good level of performance in detection accuracy and resource usage (Alsakran et al., 2020). Equally, Snort and Zeek are popular open-source IDS systems developed to enable real-time traffic analysis and protocol inspection tools with the ability to support deployment in an IoT network (Jablaoui and Liouane, 2025).

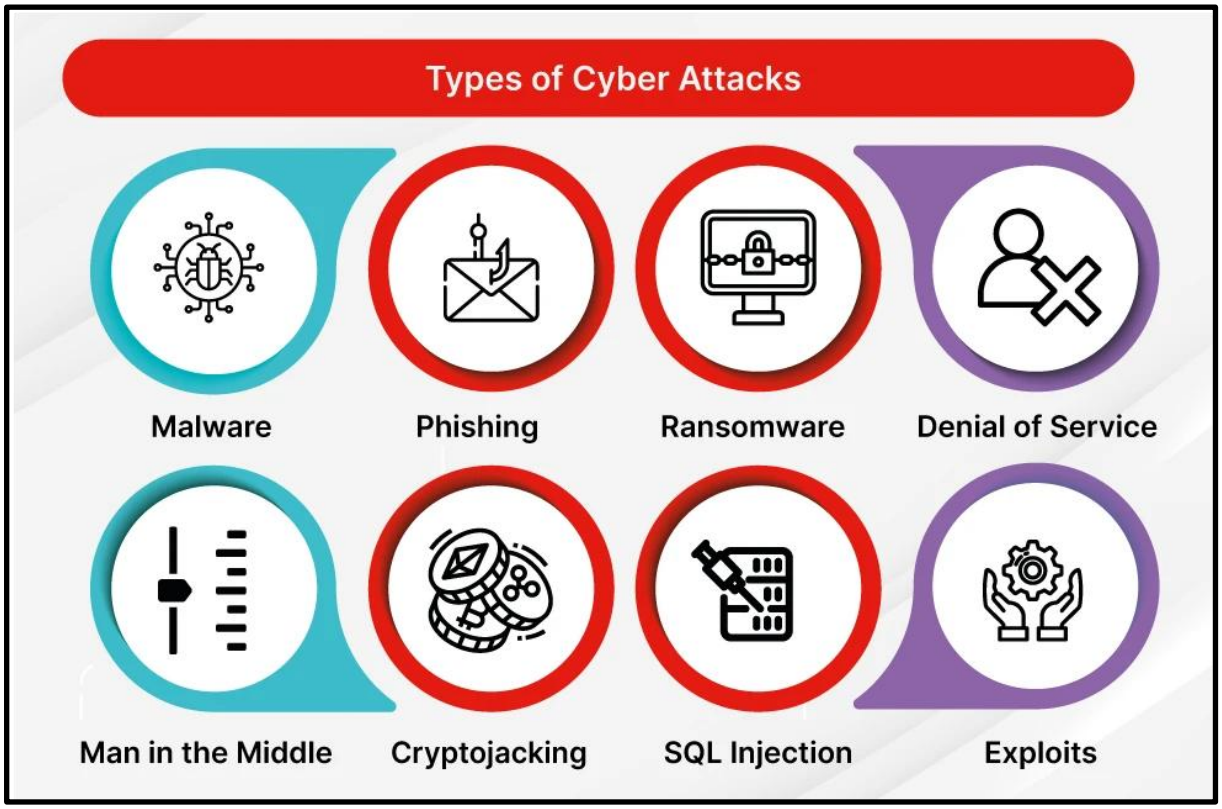


Figure 2.2: Types of intrusions in network traffic

(Source: Fortinet, 2025)

Figure 2.2 shows the typical form of malicious activities or cyberattacks, which are called intrusions. IDS seeks to detect such intrusions, determine the patterns of the attacks and counter the effects. These include ‘malware’, ‘phishing’, ‘ransomware’, ‘denial of service’ (DoS), ‘man-in-the-middle’, ‘cryptojacking’, ‘SQL injection’ and ‘exploits’ (Fortinet, 2025). Both of them are unique threats to IoT networks and should be detected differently within IDS.

However, advanced methods like deep learning are capable of increasing the performance of IDS in identifying advanced attacks and being efficient in terms of resource-constrained IoT systems.

2.3 Deep Learning Techniques for Intrusion Detection

Deep learning (DL) models are capable of predicting various kinds of intrusions based on learning the patterns in network traffic. They categorise intrusion into different groups such as ‘DoS’, ‘DDoS’, ‘man-in-the-middle-attack’, ‘phishing’, ‘malware’, ‘SQL-injection’ so on. The DL models identify anomalies, correlations and suspicious activities towards the right prediction and classification (Khan et al., 2022).

Deep learning methods have considerably improved the expertise of IDS in terms of ensuring the identification of elaborate and rising online hazards with superior precision. Such methods are especially applicable in detecting all forms of attacks intrusion including the attacks of “Denial of Service” (DoS), “Remote to local” (R2L), “User to root” (U2R), and probing attacks. “Convolutional Neural Networks” (CNNs) and “Recurrent Neural Networks” (RNNs) are mostly used in IDS to consider an analysis of “spatial and temporal characteristics” of network traffic, respectively (Zhang, Muniyandi and Qamar, 2025). As an example, CNNs are capable of automatically deriving hierarchical representations on the raw packet data and RNNs, such as “Long Short-Term Memory” (LSTM) networks, find temporal structures on temporal traffic data (Tsimenidis, Lagkas and Rantos, 2021). The combination enables the identification of advanced attack patterns that change over time.

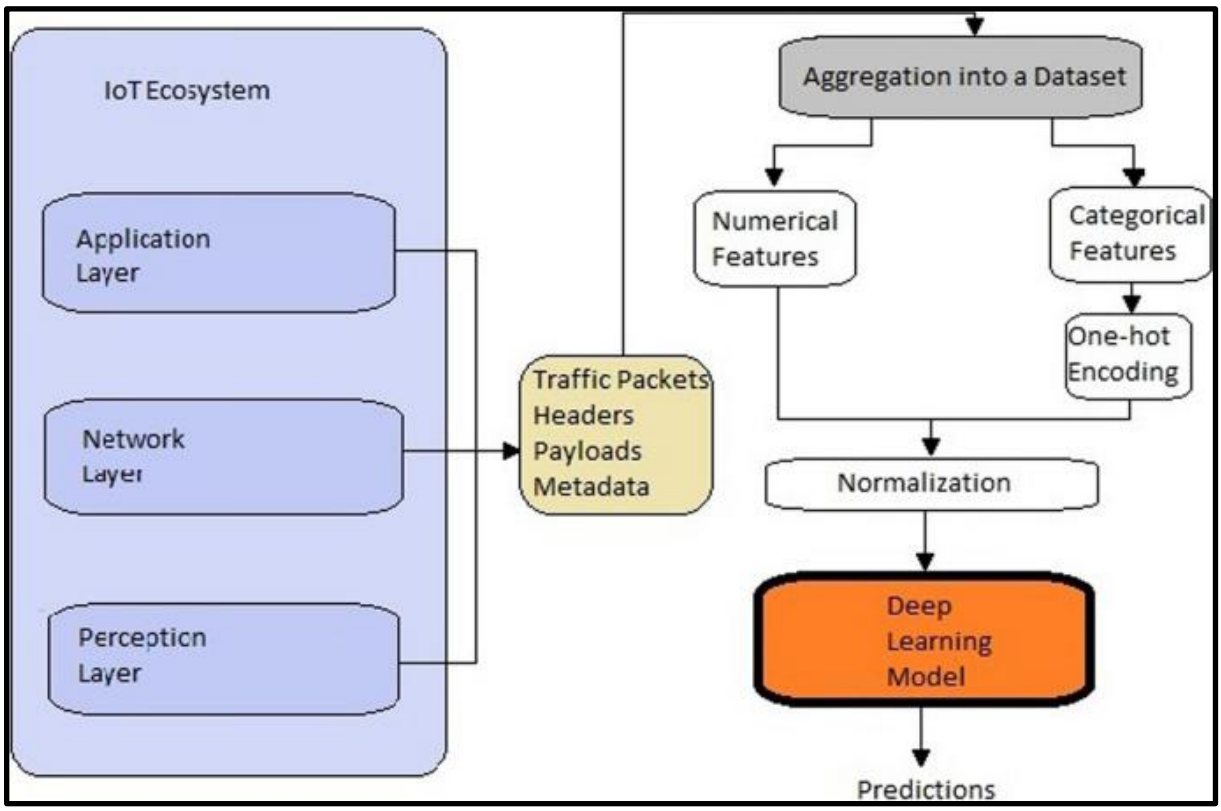


Figure 2.3: Deep learning models to enhance IDS

(Source: Tsimenidis, Lagkas and Rantos, 2021)

Figure 2.3 shows a deep learning-based IoT IDS where packets, program trafficking (headers, payloads and metadata), the network, and application levels are gathered, normalised and sent into a deep learning model to predict intrusions.

As argued by Hnamte et al., (2023), autoencoders represent a subcategory of unsupervised learning models used to create an imitation of the normal network activities. Anomalies are detected when the reconstruction error is beyond the pre-horizontal threshold, implying that possible intrusions are determined. This is specifically effective in detecting new or unseen types of attacks. As opined by Khan et al., (2022), there are hybrid systems where multiple architectures of deep learning models are merged to boost the performance of the prediction. As an illustration, one may complement CNNs with "Bidirectional LSTM networks" to extract both spatial and temporal features to improve the chance of the model detecting the complex attack situations. Moreover, "Generative Adversarial Networks" (GANs) were used to overcome the generality of training collections concerning imbalanced classes. GANs create synthetic attack samples, which balance the dataset, resulting in more successful model training and more chances to detect minority class attacks (Farhan et al., 2025).

The current trends indicate that deep learning technologies have the potential to greatly improve IDS through better accuracy, minimised false positives, and making it possible to detect new and sophisticated cyberattacks.

2.4 Performance Evaluation Metrics for Enhanced Intrusion Detection System

It is noted that IDS have benefited greatly from deep learning, with a performance that is better able to identify different vulnerabilities of intrusions using high accuracy and efficiency. Such Network traffic pattern models as CNN, LSTM networks, as well as hybrid models, among others, are found useful in detecting intricate nature in network traffic data (Celik, Basaran and Goel, 2025). These DL models are normally assessed based on input data to get statistical outputs.

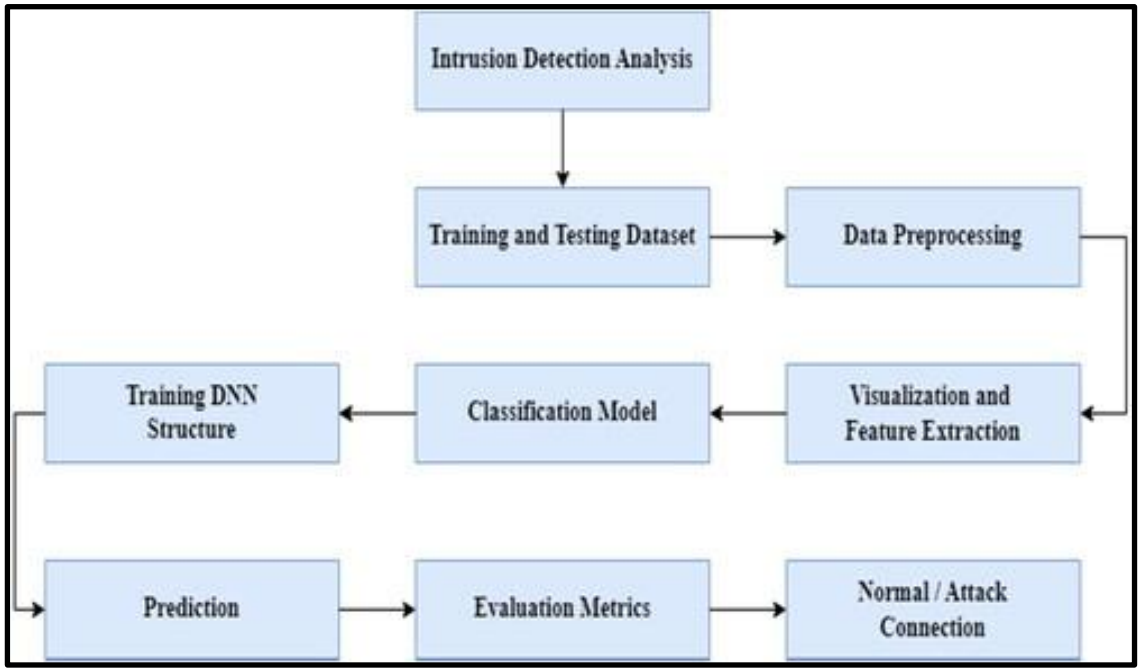


Figure 2.4: The implementation of deep learning models for intrusion prediction

(Source: Kamalakkannan et al., 2023)

Figure 2.4 illustrates how the analysis of intrusion detection works, where it begins with the preprocessing of data and retraining of a "deep neural network" (DNN) framework. It works in a sequence of classification modelling, feature extraction, prediction and evaluation based on performance measures to identify whether there is a normal connection or an attack connection (Kamalakkannan et al., 2023).

Accuracy can be used to determine the overall efficiency of the entire model prediction. The scores of precisions represent the ratio of "true positives" to all positive predications, the recall is the ability of the model to detect all the relevant instances, and the F1 score balances between the precision and recall (Shaker, Al-Musawi and Hassan, 2023). These measures play an essential role in establishing the performance of an efficient IDS, which makes the right judgments over normal and malicious human activity. In particular, an IDS implemented in CNN showed an "accuracy" of 98.59% and an "F1 score" of 84.2%, which proves its suitability in the context of real-time detection (Durgaraju and Vel, 2025).

There is also a report of the success of DL models used in multi-class classification tasks when there are multiple types of intrusions and the classification must be made. In other words, a 99.89% accuracy score was evaluated by deploying a CNN-LSTM-based hybrid deep learning architecture, signifying that it can be very impactful in detecting various types of cyberattacks (Celik, Basaran and Goel, 2025). Moreover, the DL models used in the domain of IDS have been insightful regarding identifying zero-day attacks, which investigate

a form of cyber threat that has not been studied before. Overall, DL models test 90-99 per cent scores of accuracies to predict malicious activities based on benign attacks to data sets such as “NSL-KDD” and “CICIDS 2017”, indicating the capacity to identify previously unknown patterns of intrusions (Hindy et al., 2020).

Eventually, DL models have also been major in identifying intrusions. Such AI technology also provides sensitivity in a manner to categorise cybercrimes and the possibility to generalise the performance when interacting with new variations of attacks.

2.5 Real-time Identification of Intrusions using Deep Learning Algorithms

As per the view of Almalki, (2025), in the real-time IDS enhancement, deep learning algorithms are gaining relevance because they are adaptive, correct and effective in the detection of real-time threats. These types of models, in particular, CNNs, LSTMs and hybrid networks, work effectively at defining the complex patterns of network traffic as well as identifying anomalies, which are expressions of the cyberattacks. CNNs can capture the space features, but LSTMs have the ability to capture temporal dependencies, making them both useful to detect moving threats. However, the combined “CNN-LSTM” architecture, referred to as a hybrid model, can deliver the strengths of the two models to ensure accuracy and robustness of the detection (Aljuaid and Alshamrani, 2024). In order to trace and monitor the manipulations easily and conveniently in the real-time IDS, these tools suggest the introduction of the concept of the paradigm of the graphical user interfaces (GUIs) (Bamber et al., 2025). As an example, systems like the "RealGuard" are tailored in the way to deploy the hybrid model of CNN-LSTM, which can represent the pictorialization of the real-time potential threats, the IP source and type of the intrusions utilising the assistive dashboard (Hindy et al., 2020).

Tang et al., (2020) refer to an A network intrusion detection system, as Deep Identification of Intrusion using Deep Learning Algorithms (DeepIDS) and is developed to be used in the detection of intrusion in "Software Defined Networking" (SDN). DeepIDS can provide high success rates in identifying flow-based anomalies in SDN environments via deep learning models, including fully connected deep neural networks (DNN) and gated recurrent neural networks (GRU-RNN). It also guarantees proper throughput, latency and resource utilisation performance. It establishes flow-based anomalies and offers them within a flexible, efficient, and accurate threat detection solution that does not have a crucial impact on controller performance, and results indicate an up to 90% detection accuracy (Tang et al., 2020). High-performing deep-learning-based real-time IDS by incorporation of elegant user interfaces can

considerably enhance detection, responsiveness, and user-friendliness, but they still have shortcomings in terms of scalability, explainability, and threat adaptability (Kamalakkannan et al., 2023).

These systems are deep learning and, thus, can optimise not only to identify a threat but also to offer preventive mitigation steps along with visual analytics.

2.6 Comparing Results of Previous Research

Author(s)	Models	Key findings	Limitations
(Khan et al., 2022)	DNN, ReLU, Softmax	Achieved high accuracy (99.91%) on the KDD99 dataset with low loss (0.005%) for numerical data.	Performance metrics like 'precision', 'recall', and 'F1 score' were not reported; lack of details on misclassified and classified records.
(Aljuaid and Alshamrani, 2024)	CNN	The model achieves high accuracy (99.937%–99.947%) in classifying various network traffic types, demonstrating excellent scores. It effectively detects "benign and malicious traffic" in the "CSE-CICIDS2018 dataset" scenarios.	Model performance can vary across different traffic categories, with some lower precision for specific attacks like "Label" and potential false negatives.
(Bamber et al., 2025)	CNN-LSTM	Achieved 95% accuracy, 0.89 recall, and 0.94 F1-score on the "NSL-KDD dataset", outperforming other models, offering high detection	Limited by the dataset's relevance to contemporary threats, computational complexity, and challenges with class

		performance for network traffic.	imbalance and overfitting.
(Celik, Basaran and Goel, 2025)	CNN, LSTM, RNN, GRU, AutoEncoder	Various “deep learning models”, including “CNN”, “LSTM”, and “RNN”, demonstrate high accuracy and effectiveness in detecting diverse cyber threats such as ‘DoS’, ‘DDoS’, ‘Brute Force’, ‘SQL Injection’, and ‘Botnet’, achieving accuracy scores of 99.99%.	Some models struggle with specific attack types, dataset imbalance, and computational complexity, necessitating further improvements.
(Farhan et al., 2025)	DNN with Extra Tree Classifier	Achieved 97.93% “accuracy”, “high precision”, “recall”, and “F1-score”; effective feature selection.	Limited to the “UNSW-NB15” dataset, it may not generalise across other intrusion types.
(Hindy et al., 2020)	Autoencoder & One-Class SVM	Autoencoder outperforms “One-Class SVM” in detecting complex “zero-day attacks”, with higher prediction accuracy and better handling of attack classes with varying detection patterns.	“One-Class SVM” struggles with complex attacks, and the Autoencoder's detection accuracy varies based on threshold selection. One-Class SVM lacks adaptability to non-regular attacks.
(Hnamte et al.,	DNN, CNN,	LSTM-AE outperforms	“CNN and DNN” have

2023)	LSTM-AE	DNN and CNN in 'accuracy', 'recall', 'precision', and 'F-measure'.	better training and inference times. LSTM-AE may struggle with larger datasets and higher complexity models.
(Isong, Kgote and Abu-Mahfouz, 2024)	Deep Learning-based IDS	High accuracy (up to 99.99%), detection rates (up to 99.8%), low FPRs, and reduced training time.	High computational cost, large training datasets, privacy concerns, and difficulty in adapting to new attacks.
(Kamalakkannan et al., 2023)	Combined 1-D CNN and Bi-LSTM	A 98.14% detection rate (DR), a 0.51% false positive rate (FPR), a 98% accuracy and a 0.9828 F1-score and an AUC of 0.992 for multiclass classification.	The model may experience performance degradation with larger fold sizes, and is prone to overfitting when combined with CNN and RNN.
(Shaker, Al-Musawi and Hassan, 2023)	DNN, CNN, RNN	DNN outperforms in binary classification with an accuracy of up to 98.74%, excels in NF-UNSW-NB15 and NF-BoT-IoT. CNN performs well in multiclass with an 82% F1-score. RNN shows lower performance.	DNN model requires more data for generalisation; CNN and RNN may struggle with intricate patterns and dataset imbalance.
(Zhang,	CNN-RNN Hybrid	Achieved 99.45%	Struggles with high-

Muniyandi and Qamar, 2025)	Model	accuracy on CIC-IDS2017, 99.75% detection rate on UNSW-NB15	dimensional spatial data, computationally expensive, and may have limited real-time performance due to complexity in handling both spatial and temporal features
Awajan, 2023)	Deep Neural Network-based IDS	Average accuracy: 93.74%, Precision: 93.17%, Recall: 93.82%, F1-score: 93.47%, IDS: 93.11%	Limited by realistic intrusion simulations; no access to live attackers due to security and confidentiality restrictions.

Table 1: Comparing results of existing research

2.7 Theoretical Framework

The "AttackDefense Game Theory" (ADGT) in IDS is a process where the cybercriminals and the defenders are in a competition. In the case of IDS, the security system or network manager is normally on the defensive side, guarding the security of a system, and this type of system is an evil party seeking to hack into the network (Wang et al., 2020). The game theory models the behaviour of both sides of an action and tactic with the aim of finding out what is best that each side can do in accordance with what the other is likely to conduct. The goal of the defender can be seen as to identify and prevent the attacks through proper defensive mechanisms, whereas the attacker tries to avoid the defences through novel attacking techniques.

The given theory is also closely related to the study that may be conducted on the enhanced IDS based on DL, since it would allow modelling the interactions between network intrusions and IDS. Since catastrophic attacks are to be detected and classified with deep learning models applied to network-based attacks, including CNNs, LSTMs, and RNNs, the game-theoretic approach will help envision and forecast the possible moves of the attackers (Cheng,

Fu and Sun, 2021). This has the potential of resulting in more adaptive and responsive defence systems that compete with attackers who come up with new ways to escape detectability. Using the ADGT, researchers are able to run any possible arrangement of attack-defence and refine the performance of IDS, ensuring the system works strongly and better in detecting malicious network traffic (Subba, Biswas and Karmakar, 2018).

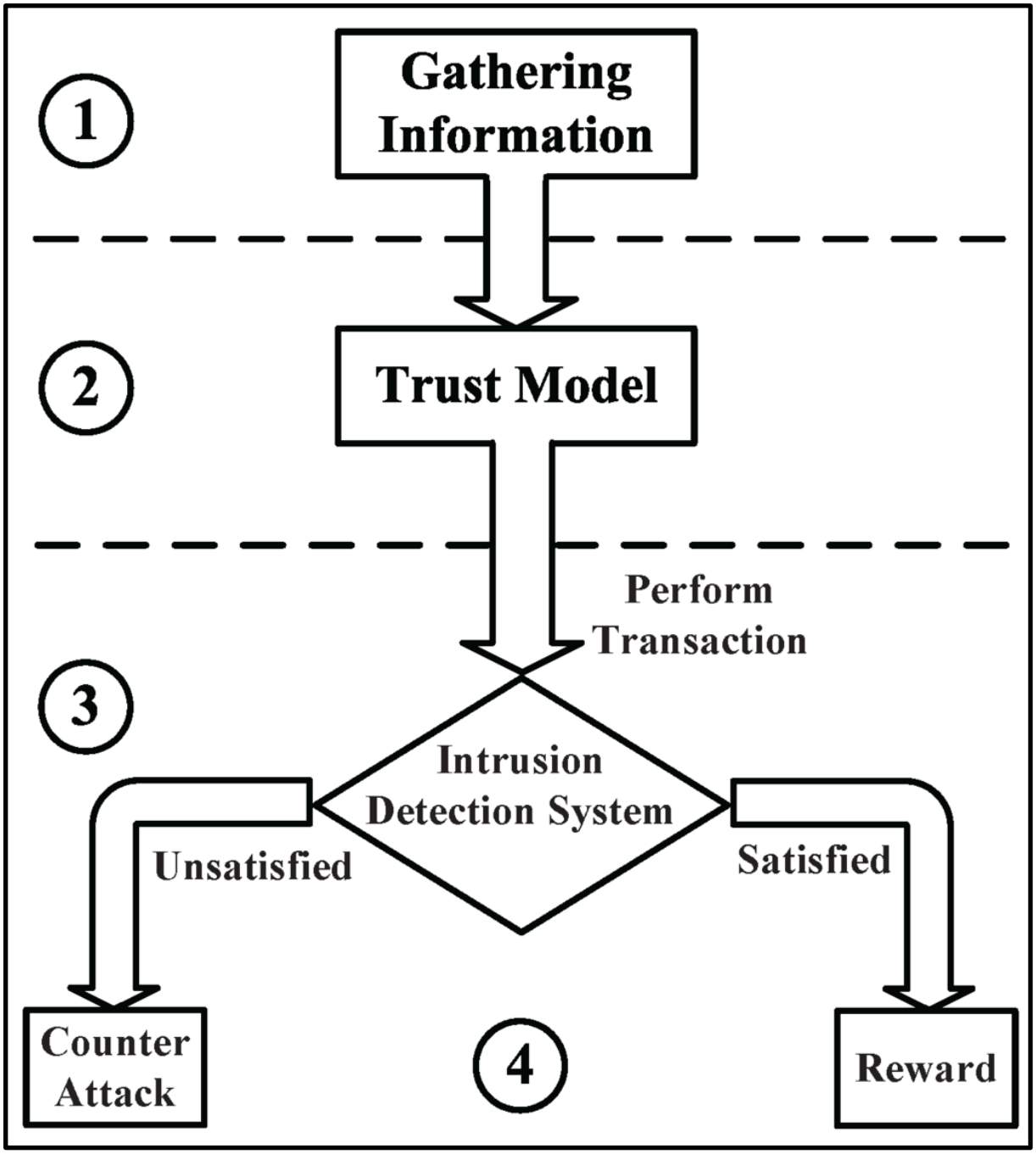


Figure 2.7: Game Theory-Based General Trust Model

(Source: Abdalzaher et al., 2016)

Figure 2.7 portrays a four-step trust model involving receiving information about the network traffic, trust model implementation, comparison of the data with an intrusion detection system and rewarding or penalising packets.

There are many benefits to using ADGT in the research. It has made the IDS system more effective since it takes into consideration the approach of the attackers and the way they can be fought back in response. The results generate not only the increased accuracy of intrusion detection but also lead to increasing the adaptability of deep learning models to the new threats (Mejdi et al., 2024). Moreover, the ADGT integration has the potential of creating more proactive IDS systems capable of learning and halting an attack in real-time and providing overall network security improvement to the IoT.

2.8 Literature Gaps

Related past investigations of this current study, based on deep learning techniques, disclose various research gaps that need to be addressed. To begin with, performance metrics used to evaluate the extent of the "precision", "recall", and "F1 scores" are not thoroughly examined in different deep learning models. Most of the studies aim at having a high accuracy rate without considering the class imbalance problem, false positives, and whether they can recognise new and sophisticated attacks. Also, the existing literature pays insufficient attention to the shortcomings of the deep learning models, such as their computational complexity, excessive training time, and the inability to adapt to real-time data.

Moreover, the hybrid architectures, such as CNN-LSTM, have yet to demonstrate their potential, but they remain limited by their scalability and overfitting related to large and complex datasets. Finally, the across-environment generalizability of the models is an issue that is under-researched, with a variety of models being trained on select datasets and failing to generalise to the actual environment.

The study fills these research gaps through the application of DL models to increase accuracy scores with effective intrusion detection in an IoT network dataset. It seeks to compare and assess deep learning architectures and address the class imbalance issue. It also aims at improving real-time detection precision, scalability, and flexibility to various IoT settings, thus getting rid of the current drawbacks.

2.9 Summary

In Chapter 2 of the project, a thorough review of "Intrusion Detection Systems" (IDS) in IoT networks with a focus on the deep learning strategies to provide maximum accuracy in cyberattack prediction. It highlights several models of IDS, both standard and deep learning-

based models, like using CNNs, using RNNs, and hybrid models such as “CNN-LSTM”. The chapter reviews to point out the efficiency of the IDS performance parameters, such as ‘precision’, ‘recall’, and ‘F1 score’. It also outlines some gaps in the past works, which encompass class imbalance, issues of computational complexities, and real-time adaptation of data. However, it is vital to find a resolution to enhance IDS IoT networks via deep learning for instantaneous identification of cyber threats.

Chapter 3: Requirements

3.1 Introduction

This chapter describes the functionality and non-functional requirements that are needed to implement and evaluate the project. Functional requirements explain the mandatory, suggested and optional functional requirements to undertake the project. Non-functional requirements specify the performance, ease of use, reliability and security requirements to guarantee a successful implementation to be utilised with user satisfaction.

3.2 Functional Requirements

Essential

- To facilitate the implementation of the data collection strategy by the use of a secondary data collection approach from a publicly available source, such as Kaggle.
- To handle the collected dataset by making a data frame for the data processing procedure of the project.
- To carry out data preprocessing, exploratory data analysis and feature engineering techniques to create deep learning models with the Python programming language.
- To implement and train deep machine learning algorithms, such as CNN, LSTM networks, and RNN, to forecast various forms of network-based cyberattacks.
- To assess deep learning algorithms to accurately classify benign and malicious traffic on networks in terms of accuracy, recall, precision, and F1-scores, as well as a confusion matrix.

Recommended

- To establish a Command Line Interface (CLI) which is user-friendly so that IDS can forecast and identify novel patterns of attacks.
- To use k-fold cross-validation to prove the performance of the models using various subsets of the dataset.
- To tune the models to ascertain that the process generates minimal fake positives, as may be necessary to make the system more flexible for network executives.
- To apply transfer learning with trained models to speed up the training and increase model performance.

Optional

- To come up with a prototype or website which can be implemented in a real network environment.

- To provide explainability (Explainable AI) methods to understand the actions of the deep models.
- To utilise the IDS system in a cloud platform such as Amazon Web Services (AWS) on a large scale and for dispersed processing.
- To develop deep architectures such as GANs (Generative Adversarial Networks) or Transformers to detect anomalies in network traffic.

3.3 Non-Functional Requirements

Non-functional requirements of the improved intrusion detection system (IDS) with deep learning are aimed at the systemic aspects of “performance”, “scalability”, and “usability”. This requires a highly available system with downtime reduced to a minimum, and the overall visibility of IoT traffic within a network. It also must be scalable to consider big datasets and high traffic volumes when the network expands, and use real-time processing with no considerable delays (Madhu et al., 2023). The system response time will need to be optimised so that alerts of intrusions are detected and processed within an acceptable time, allowing a quick response to security threats.

It should be a dependable, failure-prone mechanism and robust to adapt to unforeseeable conditions or unusual data. The security will also become paramount since the system should preserve confidential information and make sure that any illegal user is not able to gain access and manipulate the IDS (Elnakib et al., 2023). Also, the system must be easily maintained and have a code that is easily updated to add new features or boost the performance of the model. It has to be usable, allowing network administrators to engage with the system through an intuitive interface, get alerts and configure the system (Harshavardhanan et al., 2023). Lastly, the system is supposed to be receptive to different operating environments and platforms, a fact that provides flexibility and ease of incorporating into the existing infrastructure.

3.4 Summary

Chapter 3 describes the functional specifications of the improved IDS, comprising “data collection and preprocessing”, “feature engineering”, “deep learning model implementation”, and “DL evaluation of performance”. Non-functional requirements have been concerned with real-time IDS with minimal delays, scalability for handling big network traffic data, high accuracy in detecting intrusions, and strong security for managing sensitive data.

Chapter 4: Methodology

4.1 Introduction

The chapter explains how the data analysis will be carried out to develop the improved IDS by applying deep learning models. It is centred on secondary data collection to gather an open-source dataset of an IoT network, then through data preprocessing, exploratory analysis, and feature engineering. Research approach, such as quantitative, qualitative or mixed approach, has been determined here. Ethical considerations will be covered, including problems of privacy of data and security of the system.

4.2 Data Collection

The "secondary data collection approach" has been utilised in the project to find and gather a publicly available dataset based on IoT Network intrusions. The dataset is taken from Kaggle, an open-source and freely accessible data storage platform.

Data link:

"***https://www.kaggle.com/datasets/subhajournal/iotintrusion***"

The approach is selected as cost-efficient and time-saving in the access to large, labelled and diverse datasets, including historical records that are necessary to support rapid development and testing of deep learning models used in IDS. Primary data collection is both time- and resource-demanding and costly, hence not very viable in relation to already existing and rather complete secondary data.

flow_dura	Header_Le	Protocol T	Duration	Rate	Srate	Drate	fin_flag_nu	syn_flag_n	rst_flag_nu	psh_flag_f	ack_flag_f	ece_flag_f	cwr_flag_f	ack_count	syn_count	fin_count	urg_count	rst_count	HTTP	HTTPS	DNS	Telnet	SMTP	SSH	IRC
0	54	6	64	0.32981	0.32981	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
0	57.04	6.33	64	4.29056	4.29056	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	1	64	33.3968	33.3968	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.32817	76175	17	64	4642.13	4642.13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.11732	101.73	6.11	65.91	6.20221	6.20221	0	0	1	0	0	0	0	0	0	1.01	0.04	0	0.02	0	0	0	0	0	0	0
0	0	47	64	1.95412	1.95412	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1.05246	108	6	64	1.90235	1.90235	0	0	1	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0
0.14256	2322.79	6.66	79.77	493.284	493.284	0	0	0	0	0	1	0	0	0	0.02	0	1	30.41	0	1	0	0	0	0	0
0.00214	192.52	16.89	65.73	16.8832	16.8832	0	0	0	0	0	0	0	0	0.01	0	0	0.01	0.02	0	0	0	0	0	0	0
0	54.2	6	64	11.2435	11.2435	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0.22319	61.54	6.11	64.64	9.08788	9.08788	0	0	0	0	1	1	0	0	0	0	0.11	0.99	0.99	0	0	0	0	0	0	0
0	54	6	64	17.3332	17.3332	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	75.46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6.74501	108	6	64	0.29652	0.29652	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	64	1.50715	1.50715	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.01998	58.56	6.11	64.64	1.62796	1.62796	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.08833	29216	17	64	7752.32	7752.32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4.2: Data description

The dataset contains 1191264 variables of network traffic, for 47 columns that describe several IoT intrusions. These characteristics are the duration of flow, protocol, flags, traffic rates, HTTP, Telnet, DNS and so on. There is a label feature that shows the type of attacks, such as 'DDoS', 'SQL injection', 'Spoofing', 'Brute Force', 'Web-based attacks' and 'Mirai Botnet attacks'. The availability of this secondary data gives a thorough reflection of the IoT network traffic in the real world, thus making it possible to create a trustworthy IDS system (Noori Shaker et al., 2023). The range of attack types and instances makes the dataset

appropriate for developing and testing deep learning models to detect malicious traffic effectively.

4.3 Research Approach

The given study on the research approach of strengthening IDS through the application of deep learning faces a mixed-method approach that implies the use of “qualitative and quantitative methods”. The quantitative analysis refers to the usage of numbers as a means of measuring and gauging variables with an emphasis on statistical incorporation and objective findings. Qualitative analysis, in turn, examines non-numerical information, which infers the patterns, themes, and situational understanding (Chaganti et al., 2023).

Quantitative methods will be used to train and test deep learning models based on a numeric dataset obtained from the network traffic data. Accuracy, recall, precision, F1-scores and “confusion matrices” are some of the numeric performance metrics used to evaluate the models. This method is necessary in determining the values of the IDS and the ability to decide between good and bad traffic in the network under diverse circumstances (Kamil and Mohammed, 2023).

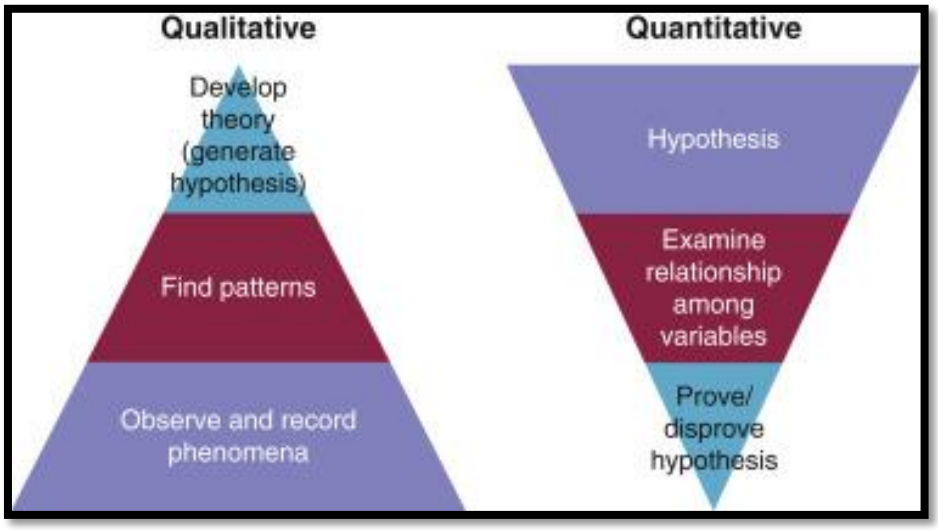


Figure 4.3: Mixed method approach of the research

(Source: Bazen, Barg and Takeshita, 2021)

Alternatively, there is also the qualitative method that will be applied to review the current literature on the current study to critically analyse limitations. The literature review with related works of the study will be compared with the findings of the study to validate the data-driven solutions. The identified literature gaps of existing studies will be resolved by quantitative findings. It also aids in giving a concept of the project to the wider environment of cybersecurity and deep learning.

The present approach lies in a mixed method, which leads to the holistic assessment of the technical and theoretical layers of the enhanced IDS.

4.4 Ethical Considerations

The ethical issues of the project are minimal due the secondary collection of the data conducted by using a publicly accessible source. There are no such ethical concerns, as the data contains historical values of IoT network intrusions. Nevertheless, it is necessary to guarantee that the data applied will be anonymised and have no personally identifiable information to respect the privacy of individuals. Also, the study must endorse the regulations and principles of fairness, which exclude the usage of biased datasets that may determine whether the deep learning models perform well or not (Alsubaei, 2025). Moreover, security and reliability should be of serious concern to the system to avert misuse or illicit entry to the analysis, where it involves the creation and protection of sensitive data on the network.

4.5 Summary

Chapter 4 defines the approach to improving IDS based on deep learning models, where CNNs, LSTMs, and RNNs will be used. It focuses on the process of gathering a publicly available IoT network intrusion dataset. The study resembles a mixed-method approach where quantitative methods of analysing models are combined with qualitative literature on the IDS development to improve its practical and theoretical aspects. Ethical concerns are minimal in this project, since the data used is free and does not include any private information. The strategy will be to maximise IDS performance in view of scalability and usability.

Chapter 5: Design

5.1 Introduction

The design section of the study describes the applicability of the Python programming language using tools such as “TensorFlow” and “Keras” to the development of three neural network models in predicting IoT network intrusions. It indicates that Google Colab will be used to execute the experiments. The chapter covers the deployment of data mining and further predictive analysis on network intrusion through classification techniques to evaluate the predictive models under specific metrics such as precision, recall, and F1 scores. This design process will be used in this project for an insightful intrusion detection process.

5.2 Research Design

Experimental research design describes an analytical method in which one or more independent variables are altered to observe the response among the dependent variables to establish a “cause-and-effect” relationship. This design is commonly employed for hypothesis testing, empirical findings, theory verification or the effectiveness of the given interventions (Ali et al., 2025).

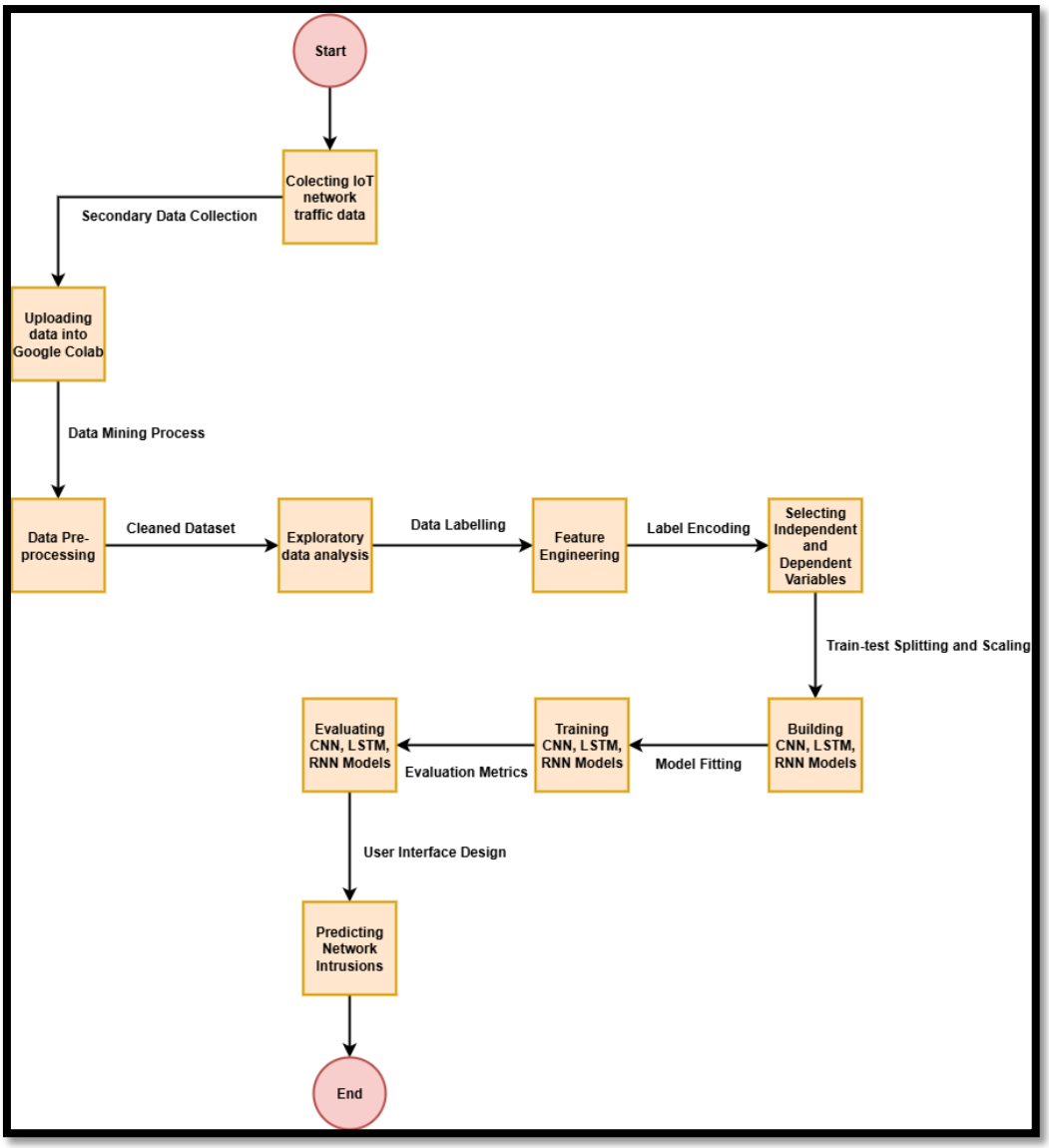


Figure 5.2: Flowchart of experimental analysis

This study employs an experimental design procedure to conduct a practical solution of data analysis using deep learning algorithms. This process is selected due to its reliable nature, practical experience and real-time analysis by making the "relationship between the independent and dependent variables". This type of data analysis will be used to practically compare the different deep learning models to detect network intrusions. It will include the "data mining" process, such as "data pre-processing" steps, "Exploratory data analysis" (EDA), and "feature engineering" (Ajagbe et al., 2024). Classification-based predictive data analytics included, in this case, is the use of multiclass target variables and training the neural network, along with testing and the evaluation metrics. Subjecting the experiment to training and testing of DL models would make the present study find out which algorithm is most effective in the classification of malicious and benign network traffic (Saranya et al., 2020).

Through this, the methodology will be in a position to empirically find out which model will be most effective in the classification of malicious traffic in a network, thus maximising intrusion detection accuracy.

5.3 Tools and Techniques

5.3.1 Tools

Programming Languages:

Python

Software Platform:

Google Colab

Libraries:

- pandas
- numpy
- scipy
- sklearn
- matplotlib
- seaborn
- tensorflow
- Joblib

```
# Insert necessary libraries
import pandas as pd
from google.colab import drive
import numpy as np
from scipy.stats import zscore
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import BatchNormalization, MaxPooling1D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import LSTM, SimpleRNN, Conv1D
from sklearn.metrics import classification_report, confusion_matrix
from tensorflow.keras import regularizers
from tensorflow.keras.layers import SpatialDropout1D
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import roc_auc_score, roc_curve
from sklearn.preprocessing import label_binarize
from tensorflow.keras.models import load_model
import joblib
import warnings
warnings.filterwarnings("ignore")
```

Figure 5.3.1: Python libraries

The above figure portrays the essential Python libraries that are imported in Google Colab before initiating the data analysis method. All the mentioned tools above are inserted into the Python Notebook significantly.

5.3.2 Techniques

```
# Mount Google Drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# Define the path to the dataset in Google Drive
file_path = '/content/drive/MyDrive/IoT_Intrusion.csv'

# Load the data and make a dataframe
intrusion_df = pd.read_csv(file_path)

# Display the first 10 values
print(intrusion_df.head(10))
```

Figure 5.3.2.1: Uploading the data into Google Colab

At first, the IoT network dataset was loaded to the Google Drive, and then it was mounted in the Google Colab by employing the “google.colab” library. After the data is successfully

mounted, the data has been loaded in Google Colab by using the “pandas” module, which converts it into a dataframe.

```
# Check the dimension of the DataFrame
print("Shape of the DataFrame:")
print(intrusion_df.shape)

# List the column names
print("\nColumn names:")
print(intrusion_df.columns.tolist())

# Get detailed info about the DataFrame
print("\nDataFrame info:")
intrusion_df.info()
```

Figure 5.3.2.2: Understanding data patterns

The Python code implies the strategy of discovering data patterns with the help of the ‘shape()’, ‘columns.tolist()’ and ‘info()’ functions. The dimension and general structure of the information are observed to determine how many rows, columns, and data types are in the dataframe.

```
# Check for missing values
print("Null values in each column:")
print(intrusion_df.isnull().sum())

# Check for duplicate values
print("\nNumber of duplicate rows:")
print(intrusion_df.duplicated().sum())
```

Figure 5.3.2.3: Checking null and duplicate values

Missing and duplicate observations are checked by employing the “isnull()” and “duplicated()” functions to detect data integrity and transparency. Then the data is subjected to data preprocessing methods to make it ready for deep learning model training.

```
# Check for outliers using the IQR method for numerical columns
print("\nOutlier detection using IQR method:")
numeric_cols = intrusion_df.select_dtypes(include=[np.number]).columns

for col in numeric_cols:
    Q1 = intrusion_df[col].quantile(0.25)
    Q3 = intrusion_df[col].quantile(0.75)
    IQR = Q3 - Q1
    outliers = intrusion_df[(intrusion_df[col] < (Q1 - 1.5 * IQR)) | (intrusion_df[col] > (Q3 + 1.5 * IQR))]
    print(f"{col}: {len(outliers)} outliers")
```

Figure 5.3.2.4: Outlier detection method

Numeric columns are also handled using the library called "numpy". By using the “scipy” library, numerical data is tested against outliers using the Interquartile Range (IQR) method, and relevant outliers are marked as candidates for elimination.

```
attack_mapping = {
    # Denial of Service (DoS) Attacks
    'DoS-TCP_Flood': 'DoS Attack',
    'DoS-UDP_Flood': 'DoS Attack',
    'DoS-SYN_Flood': 'DoS Attack',
    'DoS-HTTP_Flood': 'DoS Attack',

    # Distributed Denial of Service (DDoS) Attacks
    'DDoS-RSTFINFlood': 'DDoS Attack',
    'DDoS-ICMP_Flood': 'DDoS Attack',
    'DDoS-SynonymousIP_Flood': 'DDoS Attack',
    'DDoS-SYN_Flood': 'DDoS Attack',
    'DDoS-PSHACK_Flood': 'DDoS Attack',
    'DDoS-TCP_Flood': 'DDoS Attack',
    'DDoS-UDP_Flood': 'DDoS Attack',
    'DDoS-ACK_Fragmentation': 'DDoS Attack',
    'DDoS-ICMP_Fragmentation': 'DDoS Attack',
    'DDoS-UDP_Fragmentation': 'DDoS Attack',
    'DDoS-HTTP_Flood': 'DDoS Attack',
    'DDoS-SlowLoris': 'DDoS Attack',

    # Mirai Botnet Attacks
    'Mirai-greeth_flood': 'Mirai Botnet Attack',
    'Mirai-udpplain': 'Mirai Botnet Attack',
    'Mirai-greip_flood': 'Mirai Botnet Attack',

    # Reconnaissance Attacks
    'Recon-PortScan': 'Reconnaissance Attack',
    'Recon-OSScan': 'Reconnaissance Attack',
    'Recon-HostDiscovery': 'Reconnaissance Attack',
    'Recon-PingSweep': 'Reconnaissance Attack',

    # Injection Attacks
    'SqlInjection': 'Injection Attack',
    'CommandInjection': 'Injection Attack',

    # Web Application Attacks
    'XSS': 'Web Application Attack',
    'Uploading_Attack': 'Web Application Attack',

    # Malware & Exploit-Based Attacks
    'Backdoor_Malware': 'Malware & Exploit-Based Attack',
    'BrowserHijacking': 'Malware & Exploit-Based Attack',
```

Figure 5.3.2.5: Feature engineering

Feature engineering strategy is applied to strengthen the data by mapping different categories of intrusions in the label column into one specific type of cyberattack. The “replace()” function is used here to replace the values of cyberattacks with one specified attack.

```
# Initialize the LabelEncoder
label_encoder = LabelEncoder()

# Apply label encoding directly to the 'label' column
intrusion_df['label'] = label_encoder.fit_transform(intrusion_df['label'])

# Verify the unique encoded labels
print("Encoded labels:")
print(intrusion_df['label'].unique())
```

Figure 5.3.2.6: Label encoding

Label encoding is through the use of the “LabelEncoder” tool in the “sklearn” package, which converts categorical labels and features into a numerical form of the proper type to be used during model training.

```
# Create the scatter plot
plt.figure(figsize=(12, 7))
sns.scatterplot(data=intrusion_df, x='ack_count', y='rst_count', hue='label', palette='tab10', alpha=0.7)

# Customize the chart
plt.title('Acknowledgement Count vs Reset Count by Attack Label', fontsize=16)
plt.xlabel('ACK Count', fontsize=12)
plt.ylabel('RST Count', fontsize=12)
plt.legend(title='Label', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True)
plt.tight_layout()
plt.show()
```

Figure 5.3.2.7: Python coding to create a scatter plot

```
# Set plot style
plt.figure(figsize=(14, 7))
sns.set_style('whitegrid')

# Create the bar plot
sns.barplot(data=intrusion_df, x='TCP', y='Rate', hue='label', palette='Paired')

# Customize the chart
plt.title('Rate vs TCP with Attack Label', fontsize=16)
plt.xlabel('TCP', fontsize=12)
plt.ylabel('Rate', fontsize=12)
plt.legend(title='Attack Label', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```

Figure 5.3.2.8: Python coding to create a bar plot

```
# Select the columns
numeric_df = intrusion_df.select_dtypes(include='number')

# Compute the correlation matrix
corr_matrix = numeric_df.corr()

# Set plot size and style
plt.figure(figsize=(24, 14))
sns.set_style("white")

# Create the heatmap
sns.heatmap(
    corr_matrix,
    annot=True,
    fmt=".1f",
    cmap="coolwarm",
    square=True,
    cbar_kws={"shrink": .8},
    linewidths=0.5,
    annot_kws={"size": 8}
)

# Customize the plot
plt.title("Correlation Heatmap of Numeric Features", fontsize=18)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()
```

Figure 5.3.2.9: Python coding to create a correlation matrix heatmap

Graphical illustration of data variables has been achieved by both the use of "matplotlib" and "seaborn" tools to create different charts to gain an understanding of the relationships between the feature and types of attacks. A bar diagram and a scatter plot have been created before label encoding, and a correlation heatmap is designed only for numeric features after label encoding.

```
# Splitting features in train and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Apply the StandardScaler
scaler = StandardScaler()

# Fit and transform the training data
X_train_scaled = scaler.fit_transform(X_train)

# Transform the test data
X_test_scaled = scaler.transform(X_test)

# Print the shapes of the train-test split datasets
print(f"Training data shape: {X_train_scaled.shape}")
print(f"Testing data shape: {X_test_scaled.shape}")
```

Figure 5.3.2.10: Feature selection process with train-test split and normalisation

The ‘label’ column of the dataset has been selected as the target variable based on the independent variables. Both independent and dependent variables are divided into training and testing data with a 7:3 ratio via the use of the “train-test split” tool. Preprocessing of data consists of normalisation using the "StandardScaler" tool that will be applied to the training and test sets of independent features.

```
# Define the CNN model
cnn_model = Sequential()

# Input layer
cnn_model.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(41, 1)))  # Conv1D works on sequences of data
cnn_model.add(MaxPooling1D(pool_size=2))  # Max pooling layer

# Adding additional convolutional layers
cnn_model.add(Conv1D(filters=128, kernel_size=3, activation='relu'))
cnn_model.add(MaxPooling1D(pool_size=2))

# Add dropout for regularization
cnn_model.add(Dropout(0.2))

# Flatten the output of the convolutional layers
cnn_model.add(Flatten())

# Fully connected layer (Dense layer)
cnn_model.add(Dense(256, activation='relu'))
cnn_model.add(Dropout(0.3))

# Output layer
cnn_model.add(Dense(12, activation='softmax'))

# Compile the model with Adam optimizer
cnn_model.compile(optimizer=Adam(), loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Display the model summary
cnn_model.summary()
```

Figure 5.3.2.11: Python code design for CNN model

```
# Build the LSTM Model
lstm_model = Sequential()

# LSTM layer
lstm_model.add(LSTM(units=128, return_sequences=True, input_shape=(X_train_scaled.shape[1], 1), dropout=0.2))
lstm_model.add(BatchNormalization())

# Add more LSTM layers for deeper learning
lstm_model.add(LSTM(units=128, return_sequences=True, dropout=0.2))
lstm_model.add(BatchNormalization())

# Add more LSTM layers (optional)
lstm_model.add(LSTM(units=128, return_sequences=False, dropout=0.2))
lstm_model.add(BatchNormalization())

# Dense layer
lstm_model.add(Dense(units=64, activation='relu'))

# Output layer
lstm_model.add(Dense(units=len(np.unique(y)), activation='softmax'))

# Compile the model
lstm_model.compile(optimizer=Adam(learning_rate=0.001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Model summary
lstm_model.summary()
```

Figure 5.3.2.12: Python code design for LSTM model

```
# Define the 2D RNN model
rnn_model = Sequential()

# Input layer
rnn_model.add(LSTM(128, input_shape=(41, 1), return_sequences=True))  # LSTM layer with 128 units and output sequence
rnn_model.add(Dropout(0.2))  # Dropout for regularization

# Additional LSTM layer
rnn_model.add(LSTM(64, return_sequences=True))  # Another LSTM layer with 64 units
rnn_model.add(Dropout(0.2))  # Dropout for regularization

# Add a BatchNormalization layer
rnn_model.add(BatchNormalization())

# Flatten the output for fully connected layers
rnn_model.add(Flatten())

# Dense layer for processing the learned features
rnn_model.add(Dense(256, activation='relu'))
rnn_model.add(Dropout(0.3))  # Dropout for regularization

# Output layer
rnn_model.add(Dense(12, activation='softmax'))

# Compile the model with Adam optimizer
rnn_model.compile(optimizer=Adam(), loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Display the model summary
rnn_model.summary()
```

Figure 5.3.2.13: Python code design for RNN model

In order to implement the model, two Python packages of deep learning, "TensorFlow" and "Keras", are employed to create three “deep learning models”, such as CNNs, LSTMs and RNNs. Three “neural network architectures” encompassing CNN, LSTM and RNN models are built with various parameters and layers. CNN model begins with a “one-dimensional” (1D) “convolutional layer” (64 filters, 3 kernel size) and max-pooling, and additional convolution layers (128 filters) include dropout layers for regularisation. The LSTM architecture used is composed of layers of LSTMs with a 128-unit size, dropout method of regularisation and batch normalisation among layers. It has a “fully connected layer” and the “SoftMax layer” where the classification is done. The RNN model, which is analogous to LSTM, is composed of the LSTM units (128 and 64) and dropout, batch normalisation and a final dense layer. All models are trained using the “Adam optimiser” and “sparse categorical cross-entropy loss”, and “accuracy” is applied as an evaluation metric. The “summary()” method has been used to get an overall summary of the developed models with their layers and parameters.

```python
# Fit the CNN model on the training set
cnn_model_fitting = cnn_model.fit(
    X_train_scaled,   # Training features
    y_train,          # Training labels
    epochs=5,        # Number of epochs
    batch_size=32,    # Batch size
    validation_data=(X_test_scaled, y_test),  # Validation data
    verbose=1         # Display progress
)
```

Figure 5.3.2.14: Python code to fit the CNN model into the training sets

```python
# Reshape the data to 3D for LSTM input
X_train_scaled_reshaped = X_train_scaled.reshape(X_train_scaled.shape[0], X_train_scaled.shape[1], 1)
X_test_scaled_reshaped = X_test_scaled.reshape(X_test_scaled.shape[0], X_test_scaled.shape[1], 1)

# Fit the model on the training data
lstm_model_fitting = lstm_model.fit(X_train_scaled_reshaped, y_train,
                    epochs=5,
                    batch_size=64,
                    validation_data=(X_test_scaled_reshaped, y_test),
                    verbose=1)
```

Figure 5.3.2.15: Python code to fit the LSTM model into the training sets

```python
# Fit the RNN model on the training set
rnn_model_fitting = rnn_model.fit(
    X_train_scaled,
    y_train,
    epochs=5,
    batch_size=32,
    validation_data=(X_test_scaled, y_test),
    verbose=1
)
```

Figure 5.3.2.16: Python code to fit the RNN model into the training sets

Each model has been fitted into the training sets to classify the multiclass cyberattacks. The "k-fold cross-validation" uses subsets of the dataset to validate the model's performance. Model training entails execution of 5 epochs on CNN, LSTM, and RNN models to optimise accuracy. The scaled train and test sets of independent variables are reshaped into the appropriate shape before LSTM model training.

```python
# Predict class probabilities
cnn_pred_probs = cnn_model.predict(X_test_scaled)

# Convert predicted probabilities to class labels
cnn_pred = np.argmax(cnn_pred_probs, axis=1)

# Generate the classification report
print("Classification Report (CNN):\n")
print(classification_report(y_test, cnn_pred))
```

Figure 5.3.2.17: Python code of CNN model evaluation

```
# Generate the confusion matrix
lstm_cm = confusion_matrix(y_test, lstm_pred)

# Plot the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(lstm_cm, annot=True, fmt="d", cmap="viridis")
plt.title("Confusion Matrix (LSTM)")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.tight_layout()
plt.show()
```

Figure 5.3.2.18: Python code of confusion matrix of LSTM model

```
# Evaluate the model on the test set
loss, rnn_accuracy = rnn_model.evaluate(X_test_scaled, y_test, verbose=0)

# Print the accuracy score
print("Test Accuracy Score (RNN):", rnn_accuracy)
```

Figure 5.3.2.19: Python code to evaluate the accuracy score of the RNN model

The trained models will be assessed on test sets that produce confusion matrices and classification reports to determine performance by using the "sklearn.metrics" library. Accuracy in this study is used to measure how many of the detected intrusions and benign traffic are correctly detected against all predictions of the models. Precision tests to determine what degree the models find specific types of attacks without falsely indicating that the benign traffic is an attack. Recall determines the measure with which the models detect all the true intrusions. F1 score is a weighted mean of both the precision and recall to guarantee sound intrusion identification (Almohaimeed et al., 2025). The true positives (correctly recognised attacks), false positives (mislabeled benign traffic), true negatives (correctly labelled benign traffic), and false negatives (missed intrusions) can be visualised through a confusion matrix heatmap.

```
# Binarize the labels for multi-class ROC computation
y_test_bin = label_binarize(y_test, classes=np.unique(y_test))
cnn_pred_bin = cnn_pred_probs

# Calculate ROC curve for each class
n_classes = y_test_bin.shape[1]
fpr, tpr, roc_auc = {}, {}, {}

for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], cnn_pred_bin[:, i])
    roc_auc[i] = roc_auc_score(y_test_bin[:, i], cnn_pred_bin[:, i])

# Plot the ROC curve for CNN
plt.figure(figsize=(8, 6))
for i in range(n_classes):
    plt.plot(fpr[i], tpr[i], label=f'Class {i} ROC Curve (AUC = {roc_auc[i]:.4f})')

plt.plot([0, 1], [0, 1], linestyle='--', color='black')
plt.title('Multi-Class ROC Curve for CNN Model')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()
```

Figure 5.3.2.20: Python code to design the AUC-ROC curve of the CNN model

```
# Calculate ROC curve for each class
fpr, tpr, roc_auc = {}, {}, {}

for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], lstm_pred_probs[:, i])
    roc_auc[i] = roc_auc_score(y_test_bin[:, i], lstm_pred_probs[:, i])

# Plot the ROC curve for LSTM
plt.figure(figsize=(8, 6))
for i in range(n_classes):
    plt.plot(fpr[i], tpr[i], label=f'Class {i} ROC Curve (AUC = {roc_auc[i]:.4f})')

plt.plot([0, 1], [0, 1], linestyle='--', color='black')
plt.title('Multi-Class ROC Curve for LSTM Model')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()
```

Figure 5.3.2.21: Python code to design the AUC-ROC curve of the LSTM model

```
# Calculate ROC curve for each class
fpr, tpr, roc_auc = {}, {}, {}

for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], rnn_pred_probs[:, i])
    roc_auc[i] = roc_auc_score(y_test_bin[:, i], rnn_pred_probs[:, i])

# Plot the ROC curve for RNN
plt.figure(figsize=(8, 6))
for i in range(n_classes):
    plt.plot(fpr[i], tpr[i], label=f'Class {i} ROC Curve (AUC = {roc_auc[i]:.4f})')

plt.plot([0, 1], [0, 1], linestyle='--', color='black')
plt.title('Multi-Class ROC Curve for RNN Model')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()
```

Figure 5.3.2.22: Python code to design the AUC-ROC curve of the RNN model

AUC-ROC ("Area Under the Receiver Operating Characteristic Curve") measures the performance of the model by comparing the "true positive rate" with the "false positive rate". In three models, multi-class ROC curves are drawn, and the AUC score is computed to determine their discriminatory power. These have been performed with the help of "roc_auc_score()" and "roc_curve()" functions of the sklearn module.

```
# Save models in the native Keras format
cnn_model.save('cnn_model.keras')
lstm_model.save('lstm_model.keras')
rnn_model.save('rnn_model.keras')

# Save the scaler
joblib.dump(scaler, 'scaler.save')
```

Figure 5.3.2.23: Saving each model in the Keras format

DL models are saved with the help of the "joblib" module of Python in the Keras format. However, a "command-line interface" (CLI) is made to predict intrusion type through three DL models by generating random input values within given ranges of scaled features. It illustrates the predicted intrusion class against each of the models and thus gives insights into network security.

5.4 Summary

Chapter 5 describes how DL models will be implemented in the IoT network "intrusion detection system" (IDS) with respect to the design and the methodology. It explains how Python, Google Colab, and such libraries as 'TensorFlow', 'Keras', and 'scikit-learn' were

used to preprocess data, create features, and train models. The study is conducted on the basis of the experimental approach because several DL models are tested to determine their efficiency in classifying network intrusions. The outcomes of predictive modelling will be determined with evaluation metrics containing accuracy, ‘precision’, ‘recall’, ‘F1-score’, ‘confusion matrices’ and ‘AUC-ROC’ curves. A “user-friendly interface” will be designed to test the predictive solutions of network intrusions.

Chapter 6: Implementation and Testing

6.1 Introduction

This part of this investigation covers the implementation results, test results of data preprocessing, exploration of information (EDA), feature engineering, model assessment and the user interface design. It refers to ‘classification reports’, ‘confusion matrices’, and ‘AUC-ROC curves’, providing the assessment of deep learning models' accuracy in the context of categorising intrusions in the network. In this chapter, statistical results of the assessment of the results of the “deep learning models” in the forecasting of cyber threats in IoT network traffic have been obtained with visual solutions.

6.2 Implementation

Data preprocessing

```
Shape of the DataFrame:
(1048575, 47)

Column names:
['flow_duration', 'Header_Length', 'Protocol Type', 'Duration', 'Rate', 'Srate', 'Drate', 'fin_flag_number', 'syn_flag_number', 'rst_flag_number', 'psh_flag_number',

DataFrame info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 47 columns):
 #   Column           Non-Null Count    Dtype
---  ------           --------------    -----
 0   flow_duration    1048575 non-null  float64
 1   Header_Length    1048575 non-null  float64
 2   Protocol Type    1048575 non-null  float64
 3   Duration         1048575 non-null  float64
 4   Rate             1048575 non-null  float64
 5   Srate            1048575 non-null  float64
 6   Drate            1048575 non-null  float64
 7   fin_flag_number  1048575 non-null  int64
 8   syn_flag_number  1048575 non-null  int64
 9   rst_flag_number  1048575 non-null  int64
 10  psh_flag_number  1048575 non-null  int64
 11  ack_flag_number  1048575 non-null  int64
 12  ece_flag_number  1048575 non-null  int64
 13  cwr_flag_number  1048575 non-null  int64
 14  ack_count        1048575 non-null  float64
 15  syn_count        1048575 non-null  float64
 16  fin_count        1048575 non-null  float64
 17  urg_count        1048575 non-null  float64
 18  rst_count        1048575 non-null  float64
 19  HTTP             1048575 non-null  int64
 20  HTTPS            1048575 non-null  int64
 21  DNS              1048575 non-null  int64
```

Figure 6.2.1: Extracting the structure of the dataset

The dataset consists of 1,048,575 data records as well as 47 entities having various categorical variables and one numeric variable. Such attributes that are related to the flow duration, HTTP, DNS, a protocol type, and attack labels show an in-depth scenario of the IoT network traffic and the type of cyber threats.

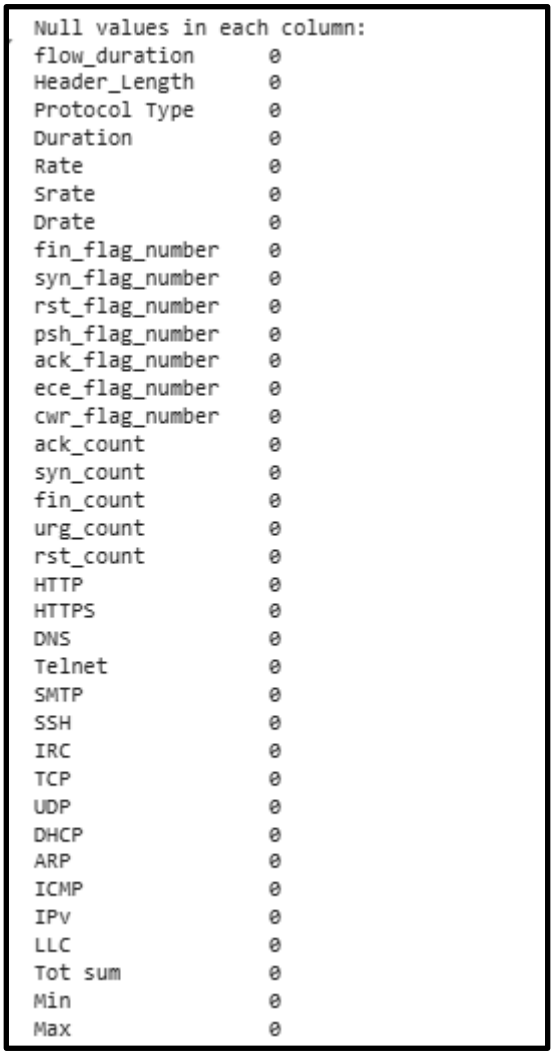


Figure 6.2.2: Data cleansing

Figure 6.2.2 highlights that there are no voids or empty rows in the dataset, ensuring a transparent and error-free dataset. Duplicates were identified in 65,556 rows that were deleted to make the dataset ready for further analysis and ensure data integrity.

```
Outlier detection using IQR method:
flow_duration: 197296 outliers
Header_Length: 257803 outliers
Protocol Type: 38682 outliers
Duration: 278566 outliers
Rate: 230651 outliers
Srate: 230651 outliers
Drate: 60 outliers
fin_flag_number: 90763 outliers
syn_flag_number: 217379 outliers
rst_flag_number: 94971 outliers
psh_flag_number: 92382 outliers
ack_flag_number: 129867 outliers
ece_flag_number: 0 outliers
cwr_flag_number: 0 outliers
ack_count: 129843 outliers
syn_count: 251313 outliers
fin_count: 139006 outliers
urg_count: 215103 outliers
rst_count: 236011 outliers
HTTP: 50840 outliers
HTTPS: 57942 outliers
DNS: 142 outliers
Telnet: 0 outliers
SMTP: 0 outliers
SSH: 63 outliers
IRC: 0 outliers
TCP: 0 outliers
UDP: 222226 outliers
DHCP: 2 outliers
ARP: 70 outliers
ICMP: 171268 outliers
IPv: 119 outliers
LLC: 119 outliers
Tot sum: 320522 outliers
Min: 300600 outliers
Max: 334942 outliers
```

Figure 6.2.3: Checking outliers in data features

The analysis identified huge outliers in several variables of the IoT network intrusion dataset. These can be attributed to their potential dominance in data integrity, including flow duration and rate that might arise during the attacks by extreme fluctuation of the network traffic. The final size of the dataset holds 983,019 records and 47 columns.

```
array(['DDoS-RSTFINFlood', 'DoS-TCP_Flood', 'DDoS-ICMP_Flood',
       'DoS-UDP_Flood', 'DoS-SYN_Flood', 'Mirai-greeth_flood',
       'DDoS-SynonymousIP_Flood', 'Mirai-udpplain', 'DDoS-SYN_Flood',
       'DDoS-PSHACK_Flood', 'DDoS-TCP_Flood', 'DDoS-UDP_Flood',
       'BenignTraffic', 'MITM-ArpSpoofing', 'DDoS-ACK_Fragmentation',
       'Mirai-greip_flood', 'DoS-HTTP_Flood', 'DDoS-ICMP_Fragmentation',
       'Recon-PortScan', 'DNS_Spoofing', 'DDoS-UDP_Fragmentation',
       'Recon-OSScan', 'XSS', 'DDoS-HTTP_Flood', 'Recon-HostDiscovery',
       'CommandInjection', 'VulnerabilityScan', 'DDoS-SlowLoris',
       'Backdoor_Malware', 'BrowserHijacking', 'DictionaryBruteForce',
       'SqlInjection', 'Recon-PingSweep', 'Uploading_Attack'],
      dtype=object)
```

Figure 6.2.4: Label columns with various intrusions

The label column keeps various types of intrusions, such as ‘DoS’, ‘DDoS’, ‘Mirai Botnet’, ‘Reconnaissance’, ‘SQL Injection’, ‘Vulnerability Scan’, ‘Benign Traffic’, and more. It is the

target column to predict cyberattacks in an IoT network environment using deep learning algorithms.

Feature Engineering

```
Updated unique labels:
['DDoS Attack' 'DoS Attack' 'Mirai Botnet Attack' 'BenignTraffic'
 'MITM-ArpSpoofing' 'Reconnaissance Attack' 'DNS_Spoofing'
 'Web Application Attack' 'Injection Attack' 'VulnerabilityScan'
 'Malware & Exploit-Based Attack' 'DictionaryBruteForce']
```

Figure 6.2.5: Applying feature engineering to the label column

In the feature engineering procedure, the label column represented minor categories of attacks in broad terms. As one example, the different categories of DDoS and DoS attacks in the label column were labelled as "DDoS Attack" and "DoS Attack", which simplified the classification and increased the model accuracy.

Exploratory Data Analysis

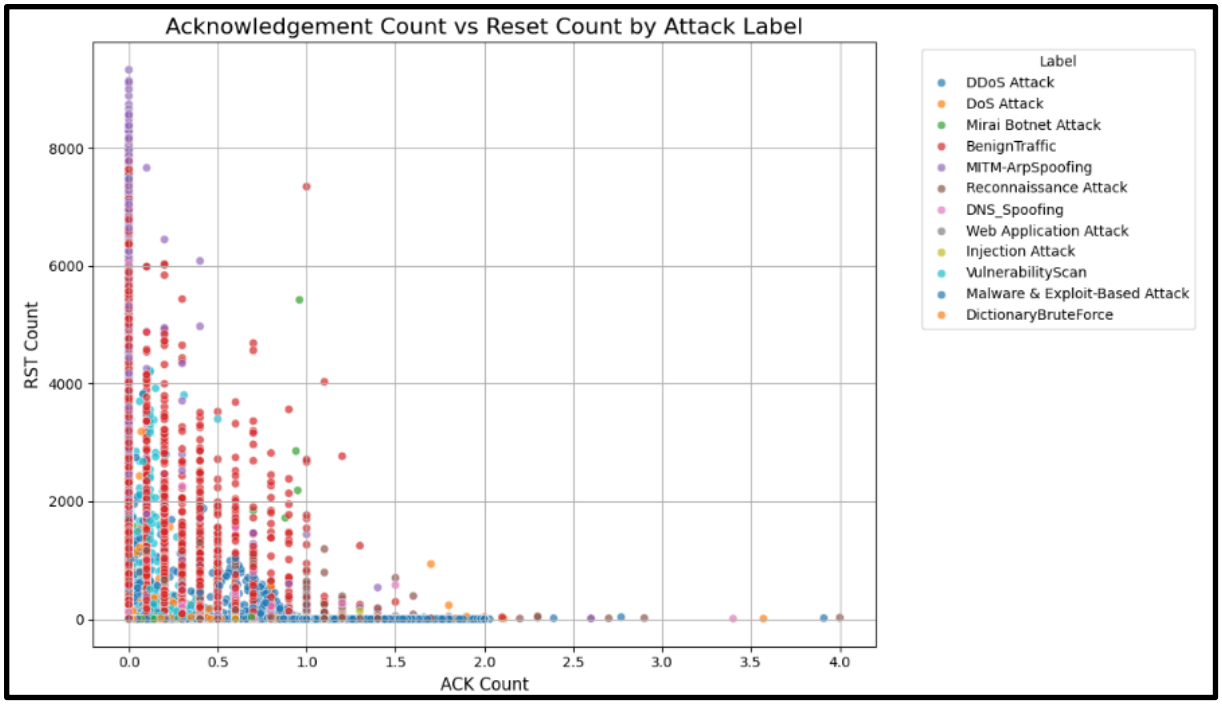


Figure 6.2.6: Scatter graph

The scatter plot indicates that the ACK and RST counts are distributed differently according to the type of attack, with DDoS attacks concentrated mostly in high values of RST counts and low values of ACK counts. Attack labels such as “BenignTraffic”, “VulnerabilityScan” have clear cluster patterns in the low ACK and the low RST counts.

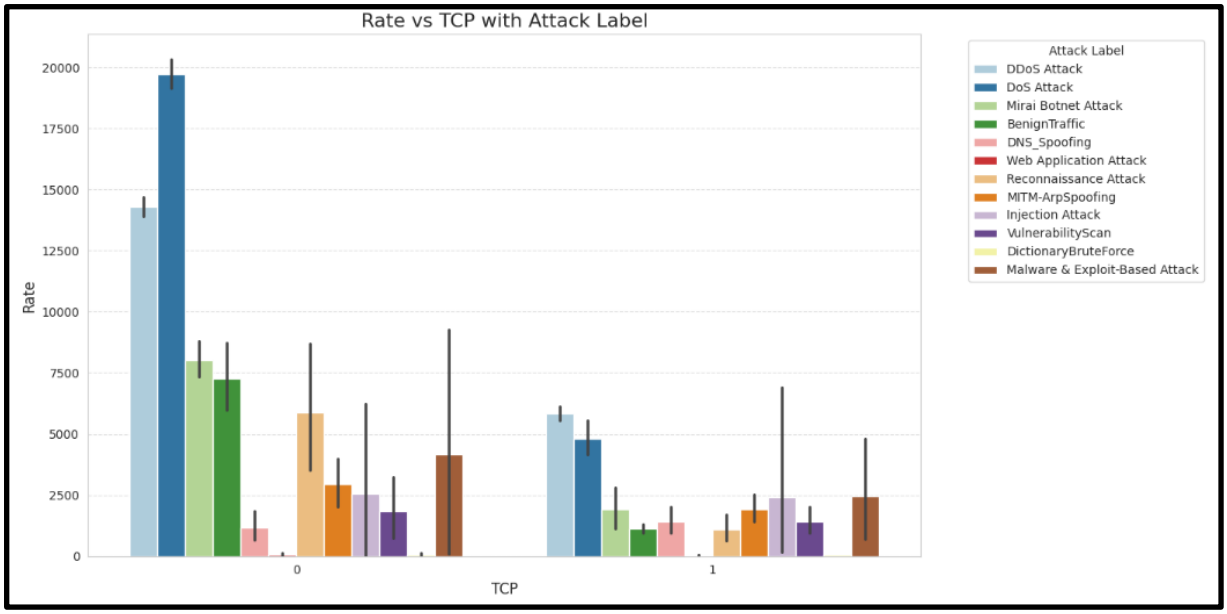


Figure 6.2.7: Bar diagram

The bar plot indicates that the highest rate is the DDoS attack, which mainly occurs in the TCP category. Other attacks, such as DoS, Mirai Botnet, and BenignTraffic, feature other rates, with lower values.

```
Encoded labels:
[ 1  4  8  0  6  9  2 11  5 10  7  3]
```

Figure 6.2.8: Encoded label features

Label encoding has been used to successfully transform the categorical values in a label column into numerical values for model training. The encoded labels were between 0 and 11, which defined the various categories of attacks to be classified.

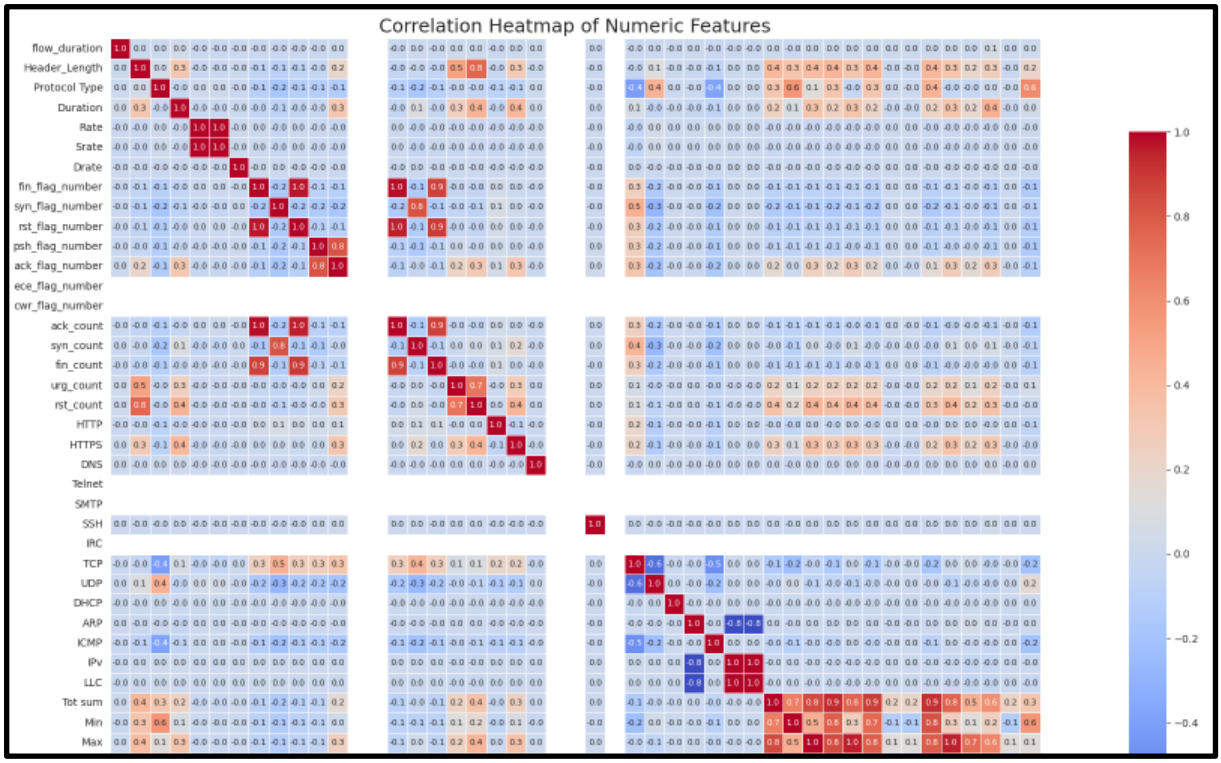


Figure 6.2.9: Correlation matrix heatmap

Through the correlation heatmap, the label column has been strongly correlated with the variables such as TCP, UDP, DHCP, and ICMP, which are important indicators for the determination of the type of intrusion. Some variables like “Telnet”, “SMTP”, “ece_flag_number”, “cwr_flag_number”, and “IRC” did not reveal any correlation to the target variable. This heatmap is drawn to select the independent variables, excluding the features which have no relationship with the label column.

```
Training data shape: (688113, 41)
Testing data shape: (294906, 41)
```

Figure 6.2.10: Shape of training and testing sets of data

The training sample is 688,113 with 41 features, and the test sample is 294,906 and likewise has 41 features. The two datasets were standardised and scaled to have consistency in the value of features when training the models. Feature scaling has been used to balance the imbalanced nature of the data features to make them ready for deep learning model implementation.

Deep Learning Building and Implementation

Model: "sequential"

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 39, 64)	256
max_pooling1d (MaxPooling1D)	(None, 19, 64)	0
conv1d_1 (Conv1D)	(None, 17, 128)	24,704
max_pooling1d_1 (MaxPooling1D)	(None, 8, 128)	0
dropout (Dropout)	(None, 8, 128)	0
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 256)	262,400
dropout_1 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 12)	3,084

Total params: 290,444 (1.11 MB)
Trainable params: 290,444 (1.11 MB)
Non-trainable params: 0 (0.00 B)

Figure 6.2.11: CNN model summary

The CNN model has 290444 total parameters, 290444 “trainable parameters” and no “non-trainable parameters”. It contains “convolutional layers”, “max-pooling”, “dropout”, “flatten” and “dense layers” and a “SoftMax output layer” to classify.

```
Epoch 1/5
21504/21504 ━━━━━━━━━━━━━━━━━━━━ 109s 5ms/step - accuracy: 0.8709 - loss: 0.3001 - val_accuracy: 0.9848 - val_loss: 0.0468
Epoch 2/5
21504/21504 ━━━━━━━━━━━━━━━━━━━━ 81s 4ms/step - accuracy: 0.9815 - loss: 0.0578 - val_accuracy: 0.9848 - val_loss: 0.0438
Epoch 3/5
21504/21504 ━━━━━━━━━━━━━━━━━━━━ 80s 4ms/step - accuracy: 0.9834 - loss: 0.0506 - val_accuracy: 0.9844 - val_loss: 0.0459
Epoch 4/5
21504/21504 ━━━━━━━━━━━━━━━━━━━━ 80s 4ms/step - accuracy: 0.9834 - loss: 0.0505 - val_accuracy: 0.9856 - val_loss: 0.0425
Epoch 5/5
21504/21504 ━━━━━━━━━━━━━━━━━━━━ 79s 4ms/step - accuracy: 0.9834 - loss: 0.0525 - val_accuracy: 0.9845 - val_loss: 0.0435
```

Figure 6.2.12: Running 5 epochs to train the CNN model

Training the CNN model over 5 epochs lowered the loss to 0.0525 compared to the previous loss of 0.3001, whilst the accuracy score rose by 8.25 percentage points to 98.34. The accuracy and the validation loss continued to improve steadily and achieved 0.0435 and 98.45%, respectively.

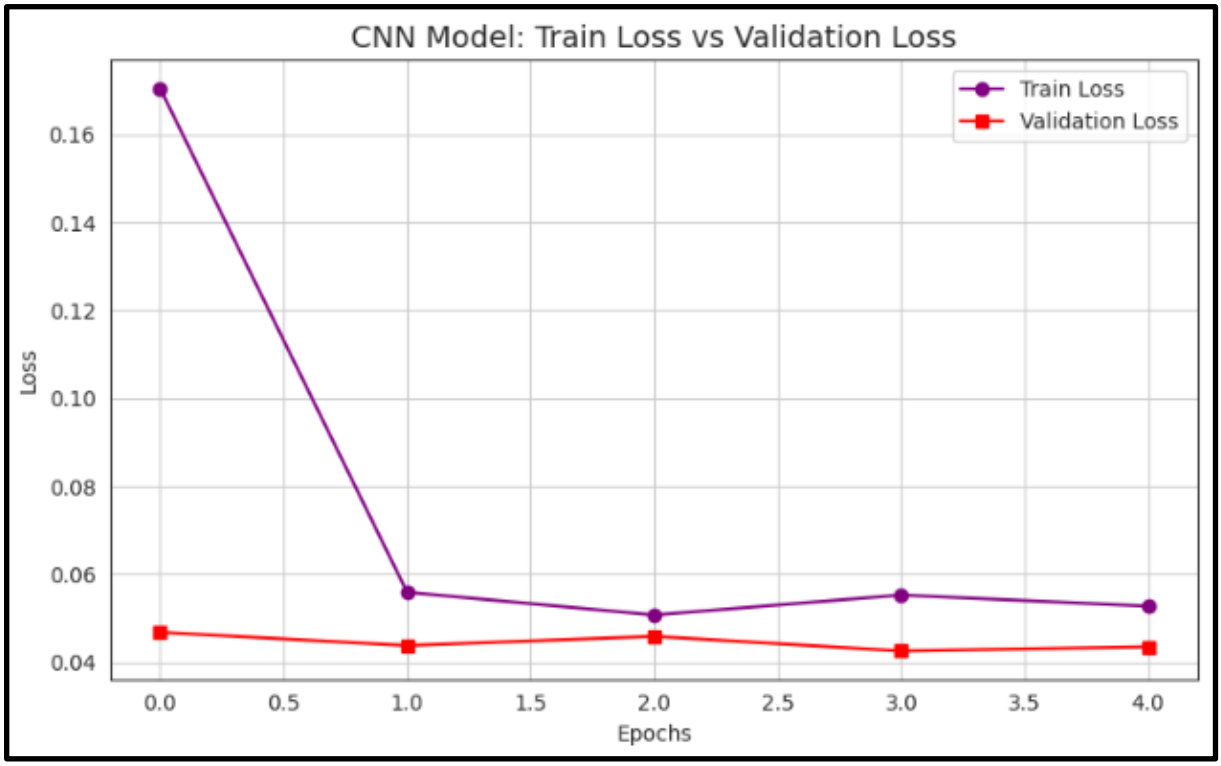


Figure 6.2.13: Line plot of train versus validation loss of CNN model

The "train versus validation loss" chart shows a decreasing behaviour, and hence, model performance is improved throughout the epochs.

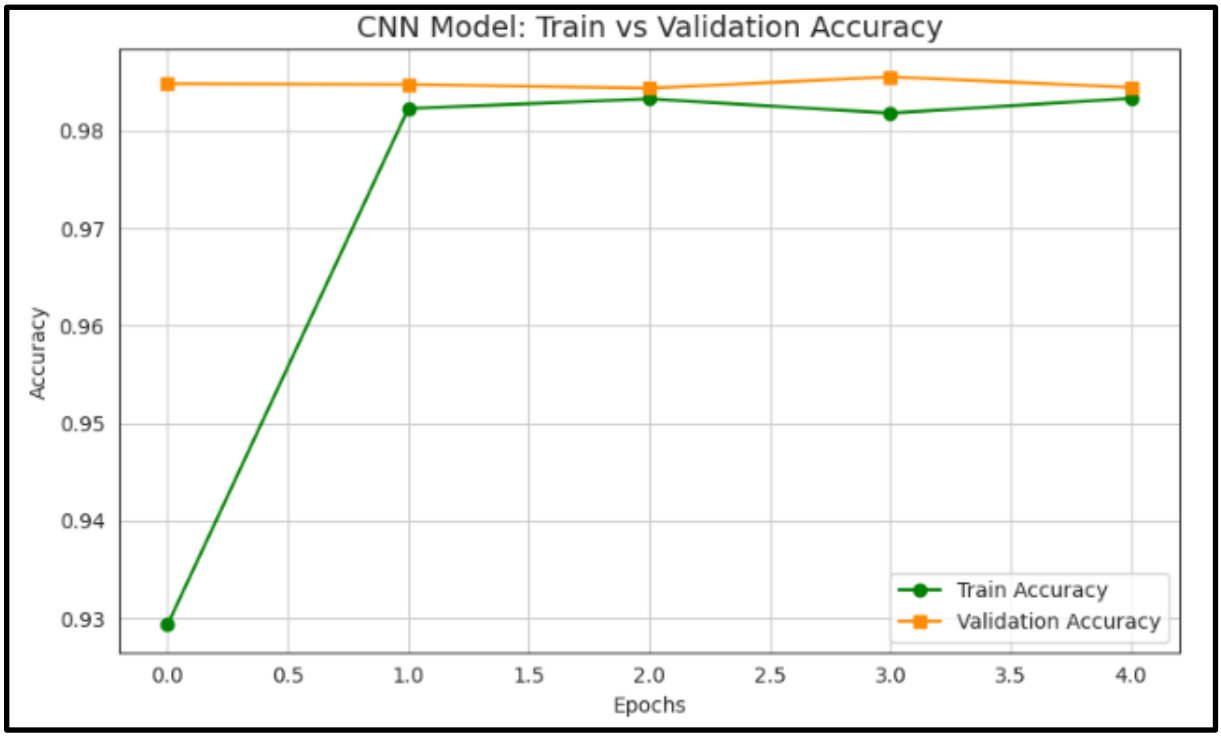


Figure 6.2.14: Train vs validation accuracy of CNN model

The "train vs validation accuracy" graph shows accuracies continuously improving, and validation accuracy closely approximating the train accuracy.

Model: "sequential_1"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 41, 128)	66,560
batch_normalization (BatchNormalization)	(None, 41, 128)	512
lstm_1 (LSTM)	(None, 41, 128)	131,584
batch_normalization_1 (BatchNormalization)	(None, 41, 128)	512
lstm_2 (LSTM)	(None, 128)	131,584
batch_normalization_2 (BatchNormalization)	(None, 128)	512
dense_2 (Dense)	(None, 64)	8,256
dense_3 (Dense)	(None, 12)	780

Total params: 340,300 (1.30 MB)
Trainable params: 339,532 (1.30 MB)
Non-trainable params: 768 (3.00 KB)

Figure 6.2.15: LSTM model summary

The LSTM model includes 340,300 overall parameters, of which 339,532 are “trainable”, and only 768 are “non-trainable” parameters. In order to provide both efficient learning of features and classification, this model makes use of LSTM layers, batch normalisation, and dense layers.

```
Epoch 1/5
10752/10752 ━━━━━━━━━━━━━━━━━━━━ 180s 16ms/step - accuracy: 0.7993 - loss: 0.5098 - val_accuracy: 0.9622 - val_loss: 0.1071
Epoch 2/5
10752/10752 ━━━━━━━━━━━━━━━━━━━━ 200s 16ms/step - accuracy: 0.9249 - loss: 0.2356 - val_accuracy: 0.9814 - val_loss: 0.0538
Epoch 3/5
10752/10752 ━━━━━━━━━━━━━━━━━━━━ 201s 16ms/step - accuracy: 0.9292 - loss: 0.2235 - val_accuracy: 0.9838 - val_loss: 0.0461
Epoch 4/5
10752/10752 ━━━━━━━━━━━━━━━━━━━━ 203s 16ms/step - accuracy: 0.9296 - loss: 0.2213 - val_accuracy: 0.9832 - val_loss: 0.0466
Epoch 5/5
10752/10752 ━━━━━━━━━━━━━━━━━━━━ 219s 18ms/step - accuracy: 0.9297 - loss: 0.2209 - val_accuracy: 0.9848 - val_loss: 0.0436
```

Figure 6.2.16: Training LSTM models with 5 epochs

In 5 epochs of training the LSTM model, the model loss ranged initially at 0.5098 and then reduced to 0.2209 whilst validation accuracy that initially stood at 96.22% improved to 98.48%, showing effective optimisation of the model.

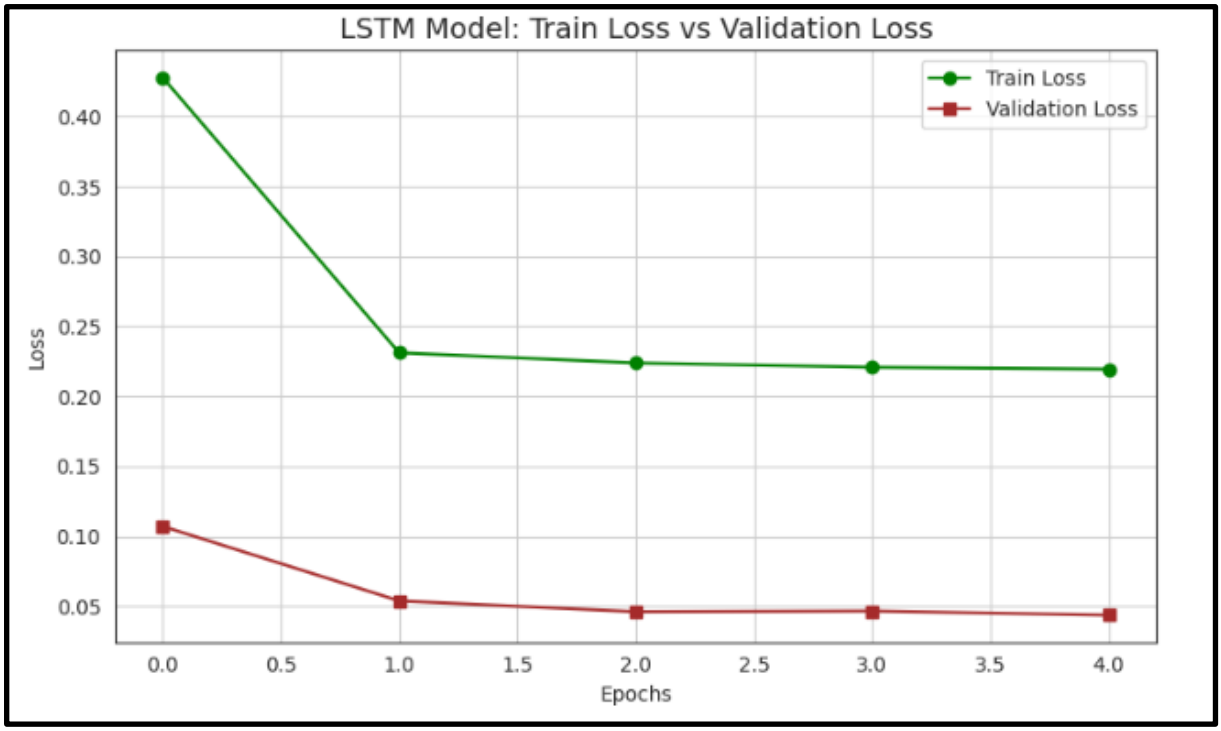


Figure 6.2.17: LSTM model train vs validation loss

This chart of the LSTM model also indicates that there is a steady decline, implying better training as the epochs proceed.

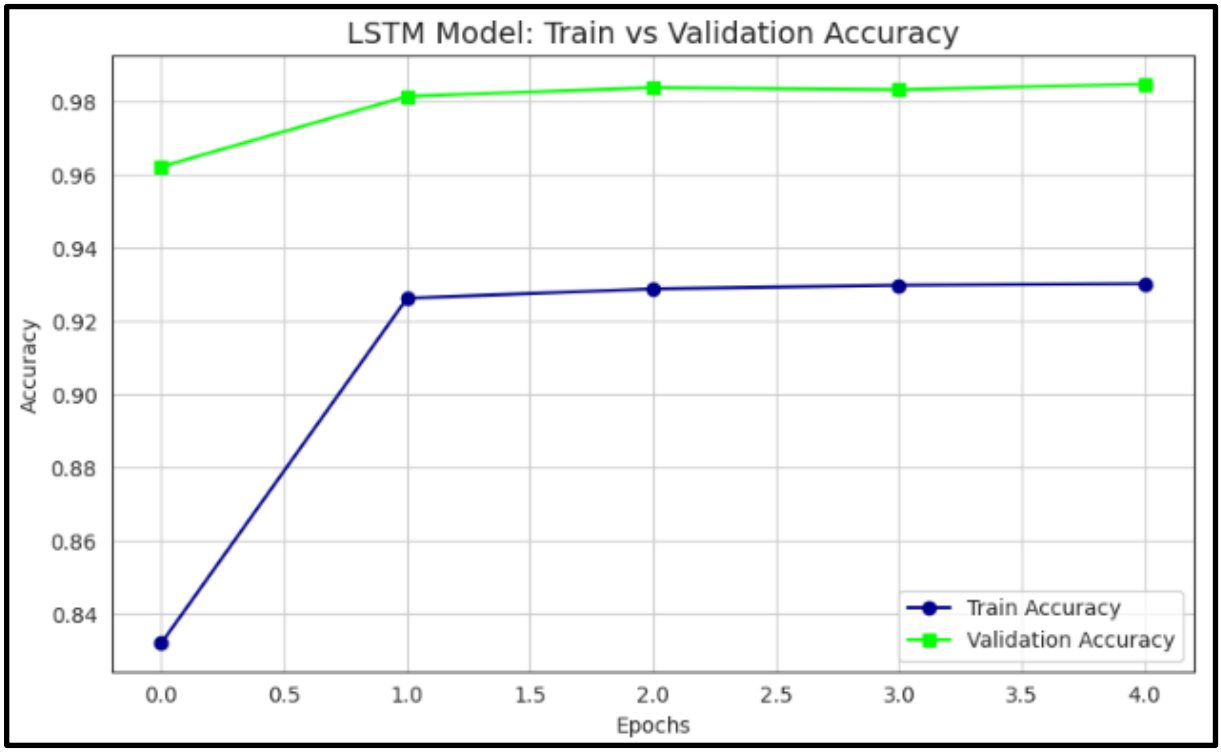


Figure 6.2.18: LSTM model train vs validation accuracy

This line graph exhibits a steady increase for both “training and validation accuracy” by running epochs, indicating a better model and better generalisation.

Model: "sequential_2"

Layer (type)	Output Shape	Param #
lstm_3 (LSTM)	(None, 41, 128)	66,560
dropout_2 (Dropout)	(None, 41, 128)	0
lstm_4 (LSTM)	(None, 41, 64)	49,408
dropout_3 (Dropout)	(None, 41, 64)	0
batch_normalization_3 (BatchNormalization)	(None, 41, 64)	256
flatten_1 (Flatten)	(None, 2624)	0
dense_4 (Dense)	(None, 256)	672,000
dropout_4 (Dropout)	(None, 256)	0
dense_5 (Dense)	(None, 12)	3,084

Total params: 791,308 (3.02 MB)
Trainable params: 791,180 (3.02 MB)
Non-trainable params: 128 (512.00 B)

Figure 6.2.19: RNN model summary

The RNN model has altogether 791,308 parameters, out of which 791,180 are trained and 128 are non-trained. It comprises various LSTM blocks, “dropout layers” to regularise them and a “dense layer” to process the features acquired.

```
Epoch 1/5
21504/21504 ━━━━━━━━━━━━━━━━━━━━ 253s 12ms/step - accuracy: 0.8228 - loss: 0.3863 - val_accuracy: 0.8593 - val_loss: 0.3180
Epoch 2/5
21504/21504 ━━━━━━━━━━━━━━━━━━━━ 260s 12ms/step - accuracy: 0.8674 - loss: 0.3067 - val_accuracy: 0.9644 - val_loss: 0.1158
Epoch 3/5
21504/21504 ━━━━━━━━━━━━━━━━━━━━ 260s 12ms/step - accuracy: 0.9612 - loss: 0.1072 - val_accuracy: 0.9824 - val_loss: 0.0481
Epoch 4/5
21504/21504 ━━━━━━━━━━━━━━━━━━━━ 259s 11ms/step - accuracy: 0.9771 - loss: 0.0627 - val_accuracy: 0.9856 - val_loss: 0.0407
Epoch 5/5
21504/21504 ━━━━━━━━━━━━━━━━━━━━ 265s 12ms/step - accuracy: 0.9830 - loss: 0.0483 - val_accuracy: 0.9855 - val_loss: 0.0409
```

Figure 6.2.20: RNN model training by running epochs

The training of the RNN model took place during five epochs with a reduction of loss (0.3863 to 0.0483) and an increase of validation accuracy (0.8593 to 0.9855).

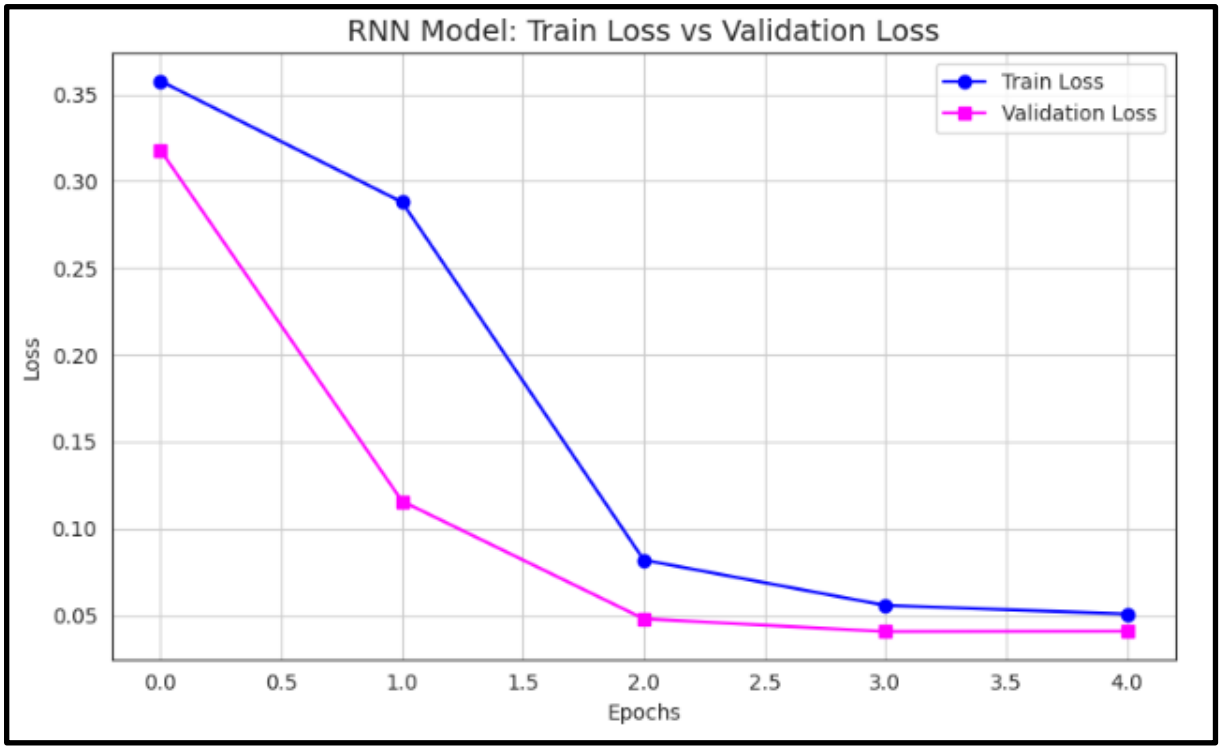


Figure 6.2.21: Line chart of RNN model train and validation loss

The line chart of "train vs. validation loss" of the RNN model depicts a gradual reduction of both "train and validation loss" and shows that the model is learning through the epochs.

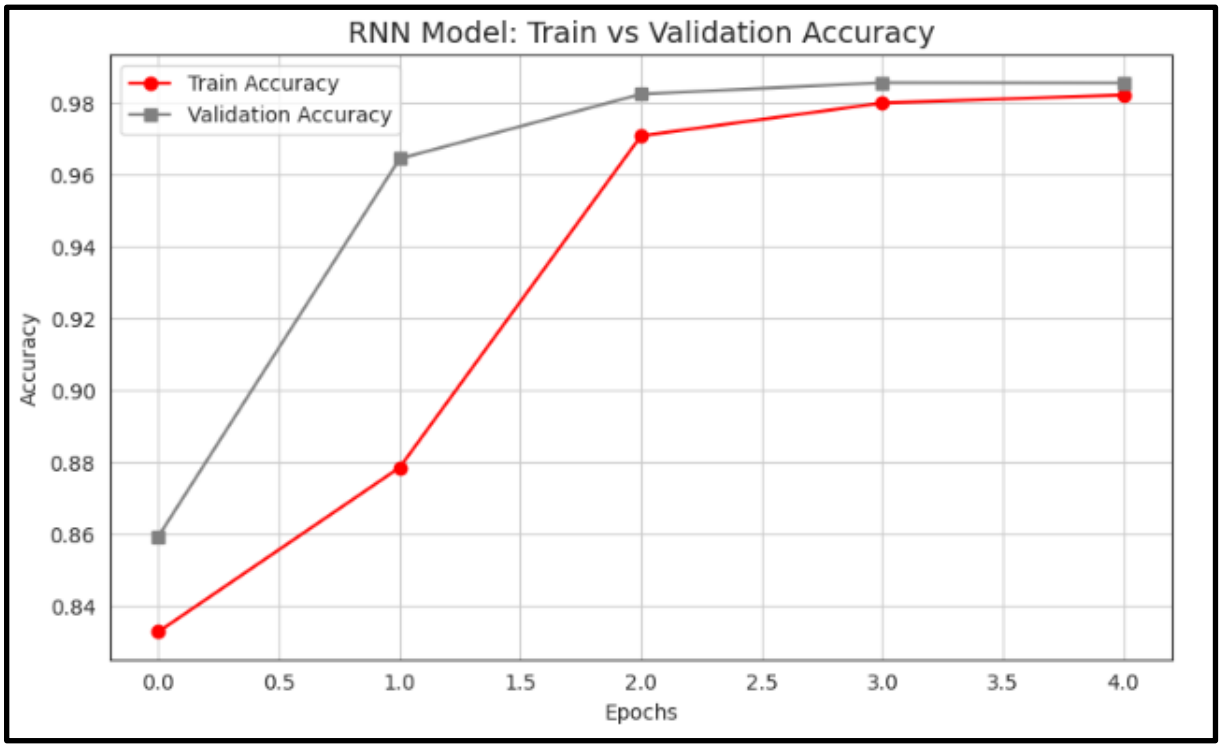


Figure 6.2.22: Line chart of RNN model train and validation accuracy

The line chart of "train vs validation accuracy" of RNN model showed clear increases in validation accuracy, the last epoch surpassing that of training accuracy.

6.3 Testing

Classification Report (CNN):

	precision	recall	f1-score	support
0	0.70	0.98	0.82	7325
1	1.00	1.00	1.00	210670
2	0.52	0.18	0.27	1213
3	0.00	0.00	0.00	92
4	1.00	0.99	0.99	53287
5	0.00	0.00	0.00	70
6	0.76	0.44	0.56	2111
7	0.00	0.00	0.00	74
8	1.00	0.99	1.00	17579
9	0.85	0.46	0.60	2203
10	0.89	0.54	0.67	257
11	0.00	0.00	0.00	25
accuracy			0.98	294906
macro avg	0.56	0.47	0.49	294906
weighted avg	0.98	0.98	0.98	294906

Figure 6.3.1: CNN model evaluation

The CNN model obtained a total accuracy of 98.45% in predicting the intrusion network. Across all types of attacks, there was a performance difference in “precision”, “recall”, and “F1-scores” of each type of intrusion. The most successful performance occurred in classifying the “DoS Attack”, “BenignTraffic”, “Mirai Botnet Attack”, and “DDoS Attack” categories, with all metrics close to 100%. Some attack types, such as “Web Application Attack”, “Injection Attack”, and “DictionaryBruteForce”, performed less well with lower scores of evaluation metrics.

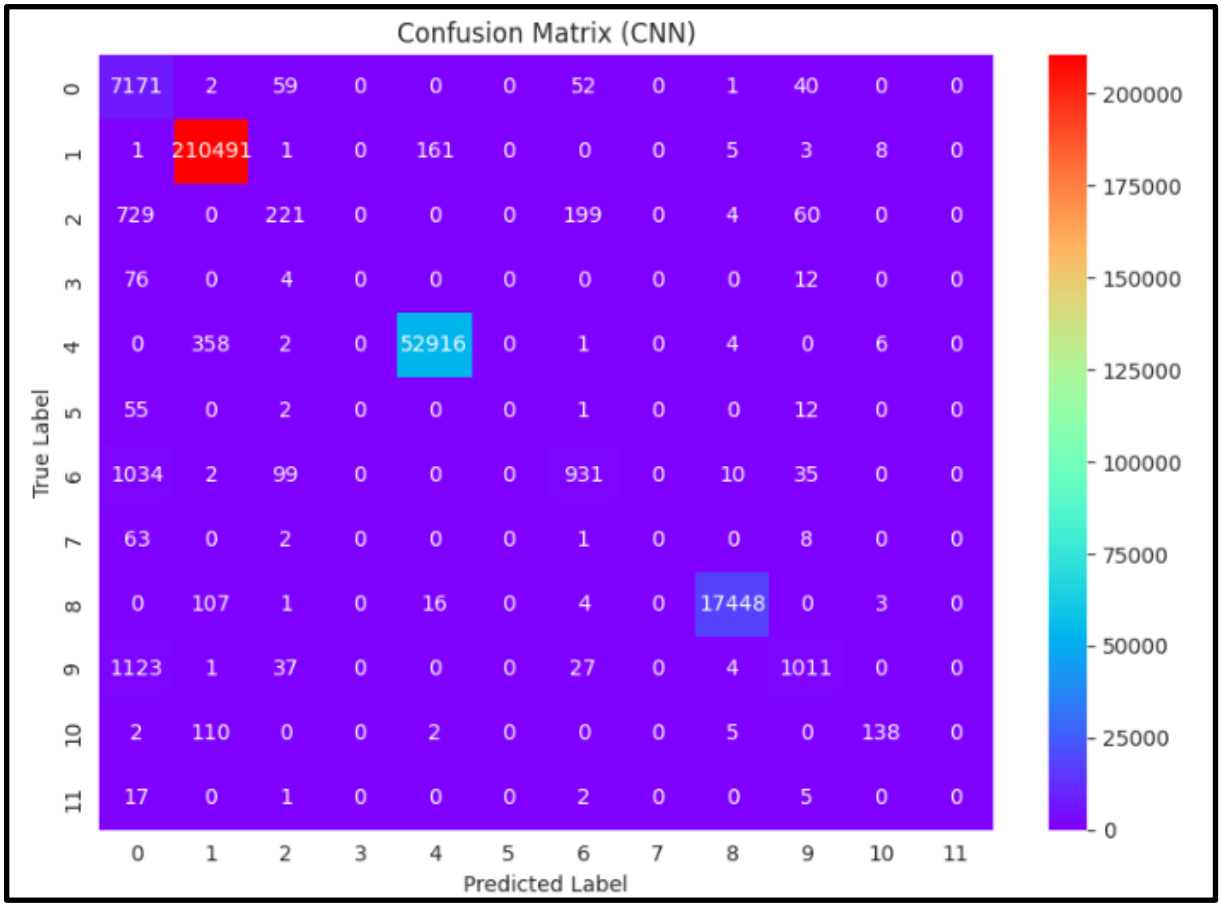


Figure 6.3.2: Confusion matrix of CNN model

The confusion matrix of the CNN model shows the “true positives” (TP) of the three categories, including “DoS Attack”, “DDoS Attack”, and “Mirai Botnet Attack”. Nevertheless, “false positives” (FP) can be noted on predictions of other attacks, especially of intrusions like “Malware & Exploit-Based Attack”, “Injection Attack”, which results in a failure to identify the true level of the attack. There are no such “false negatives” (FN) in any attack types. However, “Benign traffic” is characterised by high “True negatives” (TN), thus a good performance in the placement of non-attacks.

Classification Report (LSTM):

	precision	recall	f1-score	support
0	0.74	0.97	0.84	7325
1	1.00	1.00	1.00	210670
2	0.49	0.22	0.31	1213
3	1.00	0.13	0.23	92
4	0.99	0.99	0.99	53287
5	0.00	0.00	0.00	70
6	0.83	0.49	0.62	2111
7	0.00	0.00	0.00	74
8	1.00	0.99	0.99	17579
9	0.74	0.55	0.63	2203
10	0.86	0.49	0.62	257
11	0.00	0.00	0.00	25
accuracy			0.98	294906
macro avg	0.64	0.49	0.52	294906
weighted avg	0.98	0.98	0.98	294906

Figure 6.3.3: Classification report of the LSTM model

The LSTM model was able to attain an accuracy score that was 98.48% in the predictions of network intrusions. Different types of attack had varying “precision”, “recall” and “F1 scores”, with their top performers being the “Mirai Botnet Attack”, “BenignTraffic”, “DoS Attack” and the “DDoS Attack”. The precision, F1 score and recall of “Malware & Exploit-Based Attack”, “DictionaryBruteForce”, “DNS_Spoofing”, “Injection Attack” and “VulnerabilityScan” were insufficient, which signifies that there is still an opportunity to increase performance related to these categories.

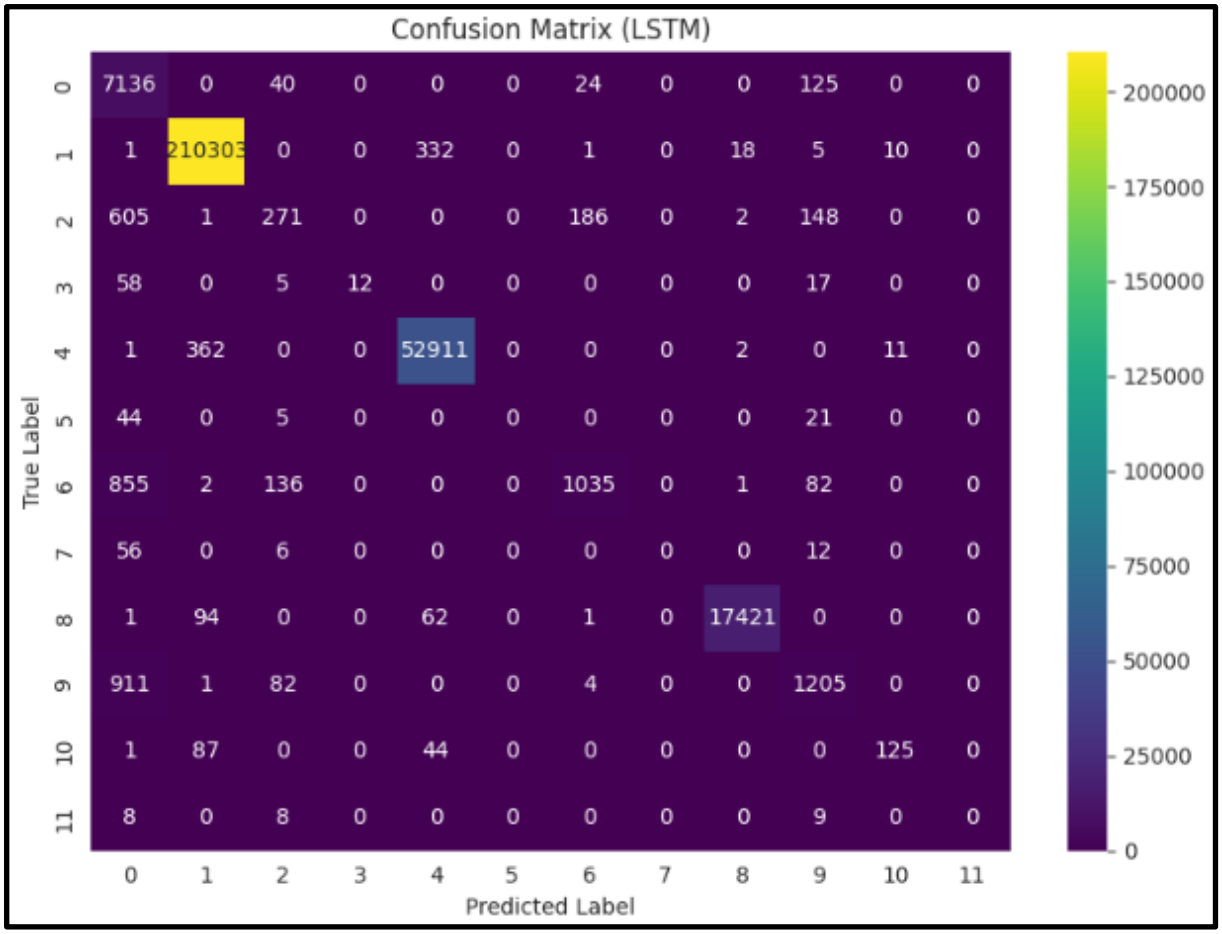


Figure 6.3.4: Confusion matrix heatmap of LSTM model

The confusion matrix heatmap of the LSTM model indicates the level of correctly predicted types of attacks. High diagonal values explore proper estimates, notably in the event of 'DDoS Attack' and 'DoS Attack'. The smaller figures of less common categories, such as “MITM-Arp spoofing”, indicate misclassifications, revealing the areas where the model can be improved, especially on minority attack categories.

Classification Report (RNN):

	precision	recall	f1-score	support
0	0.76	0.97	0.86	7325
1	1.00	1.00	1.00	210670
2	0.75	0.15	0.25	1213
3	1.00	0.17	0.30	92
4	1.00	0.99	0.99	53287
5	0.00	0.00	0.00	70
6	0.84	0.58	0.69	2111
7	0.00	0.00	0.00	74
8	1.00	0.99	1.00	17579
9	0.69	0.65	0.67	2203
10	0.91	0.57	0.70	257
11	0.00	0.00	0.00	25
accuracy			0.99	294906
macro avg	0.66	0.51	0.54	294906
weighted avg	0.99	0.99	0.98	294906

Figure 6.3.5: RNN model evaluation metrics

The RNN model attained an accuracy of 98.55%. However, precision, recall and F1 values were high in cases of “DoS Attack” (“precision 1.00, recall 1.00, F1 score 1.00”), “DDoS Attack” (“precision 1.00, recall 0.99, F1 0.99”) and “Mirai Botnet Attack” (“precision 1.00, recall 0.99, F1 1.00”). The model effectively predicted the “BenignTraffic”, indicating the malicious activity in the IoT traffic is normal. The model achieved good results in the prediction of these types of attacks and delivered effective intrusion recognition metrics.

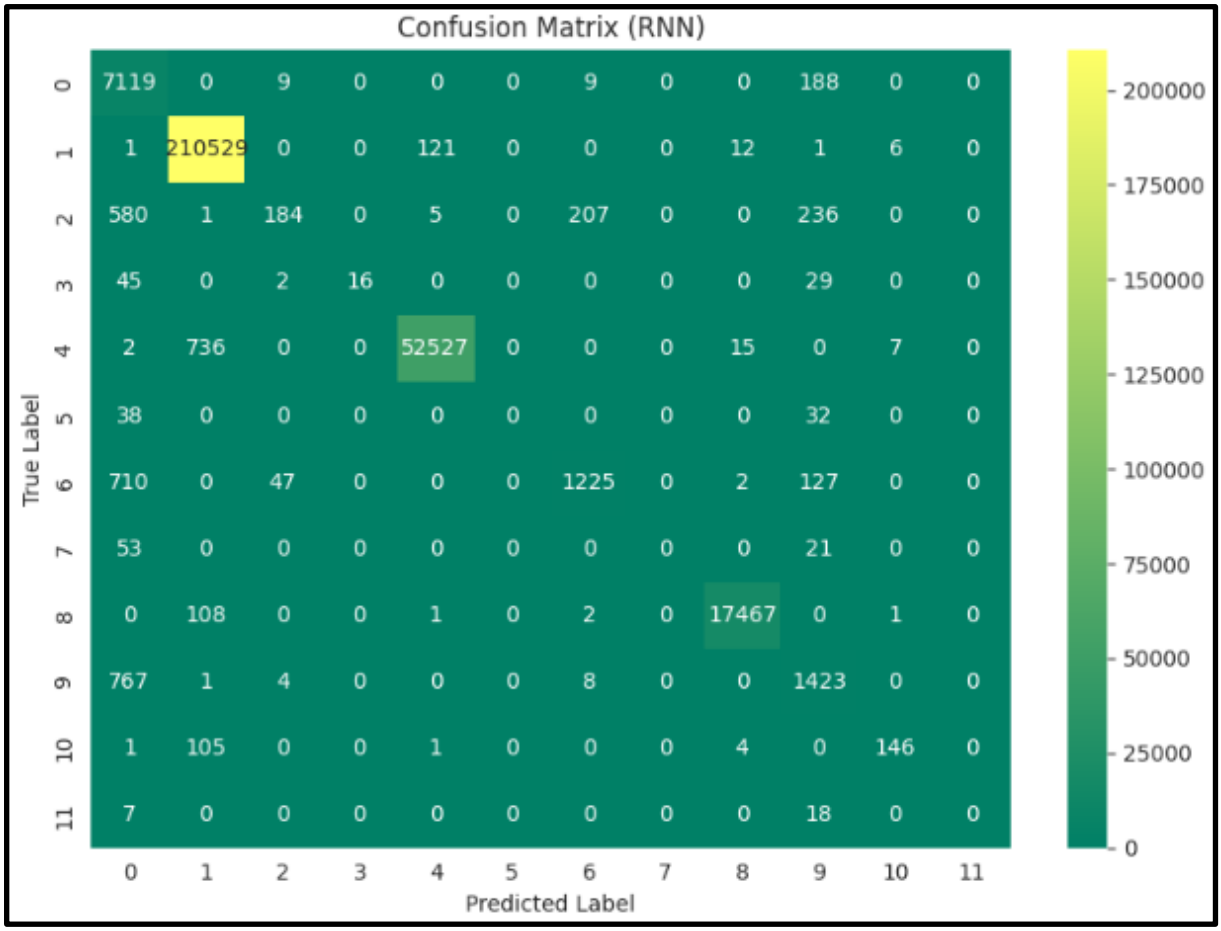


Figure 6.3.6: Confusion matrix of RNN model

The confusion matrix heatmap of the RNN model demonstrates the impact of the model in making predictions with a high accuracy for some attack types, specifically, “DoS Attack” and “DDoS Attack”. Nonetheless, the model performs poorly on the rare categories, such as “Web Application Attack”, “Injection Attack”, and “Malware & Exploit-Based Attack”, often misclassified. The matrix indicates the overall accuracy of the model of correctly using common attack labels, as well as areas to be improved, namely, a higher precision and recall of minority attack types.

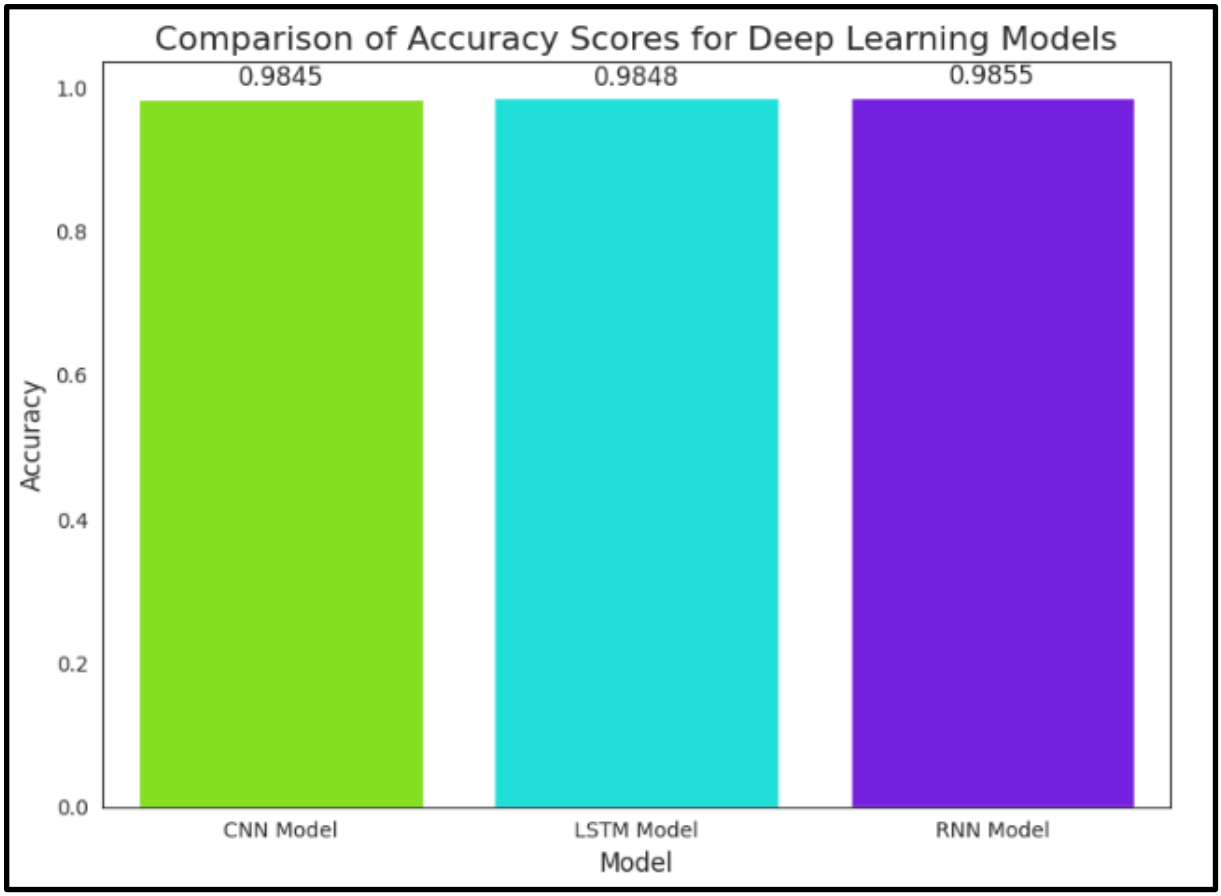


Figure 6.3.7: Bar plot comparing each DL model by accuracy scores

Regarding the bar plot of the accuracy scores, the RNN model outperforms both the CNN and LSTM models with an estimated accuracy of 0.9855, which is slightly higher than that of the CNN (0.9845) and LSTM (0.9848) models. This excellent performance can probably be attributed to RNN's ability to process the sequential data more optimally, resulting in higher results in predicting network intrusions.

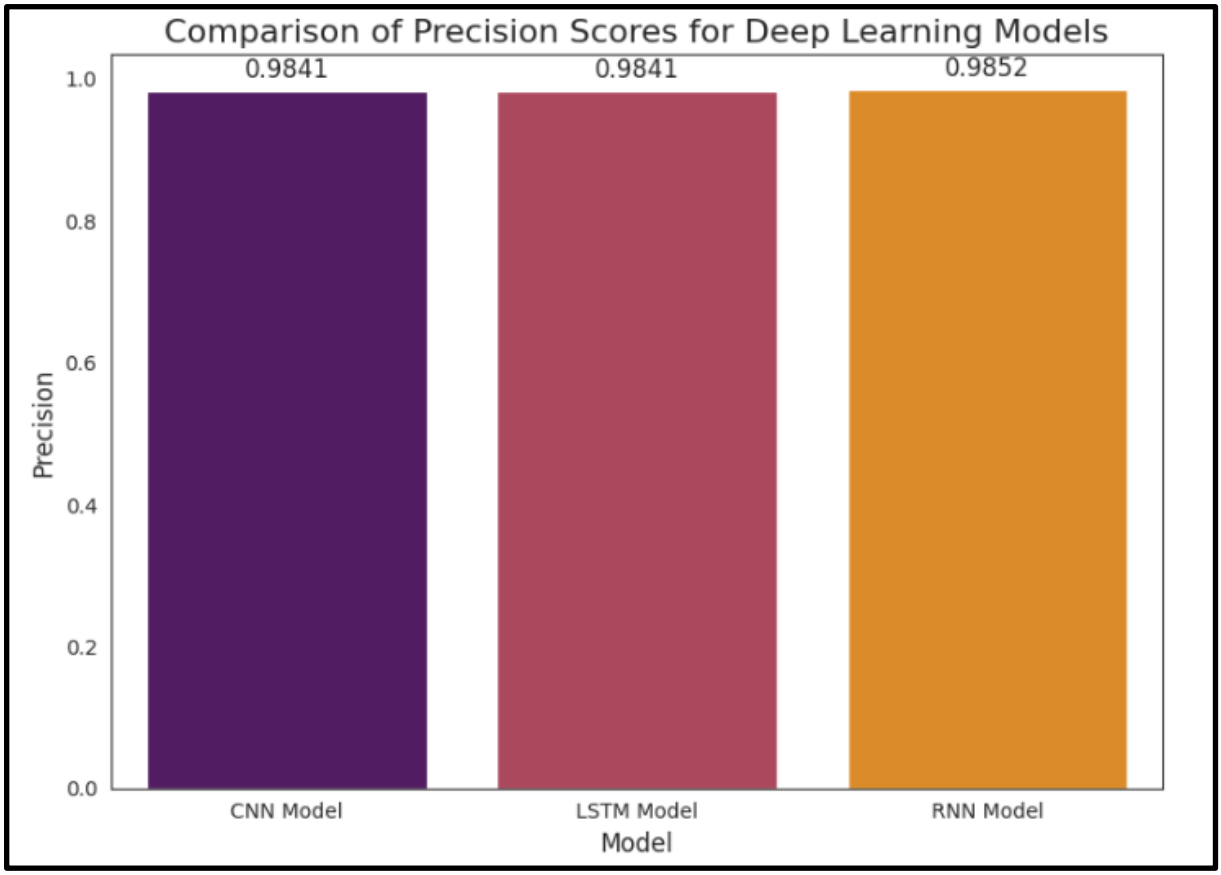


Figure 6.3.8: Bar diagram to compare all models by precision scores

It is noticed that the RNN model gave the highest precision score of 0.9852, as indicated by the bar chart. The CNN and LSTM models were very similar, with precision scores of 0.9841 in each.

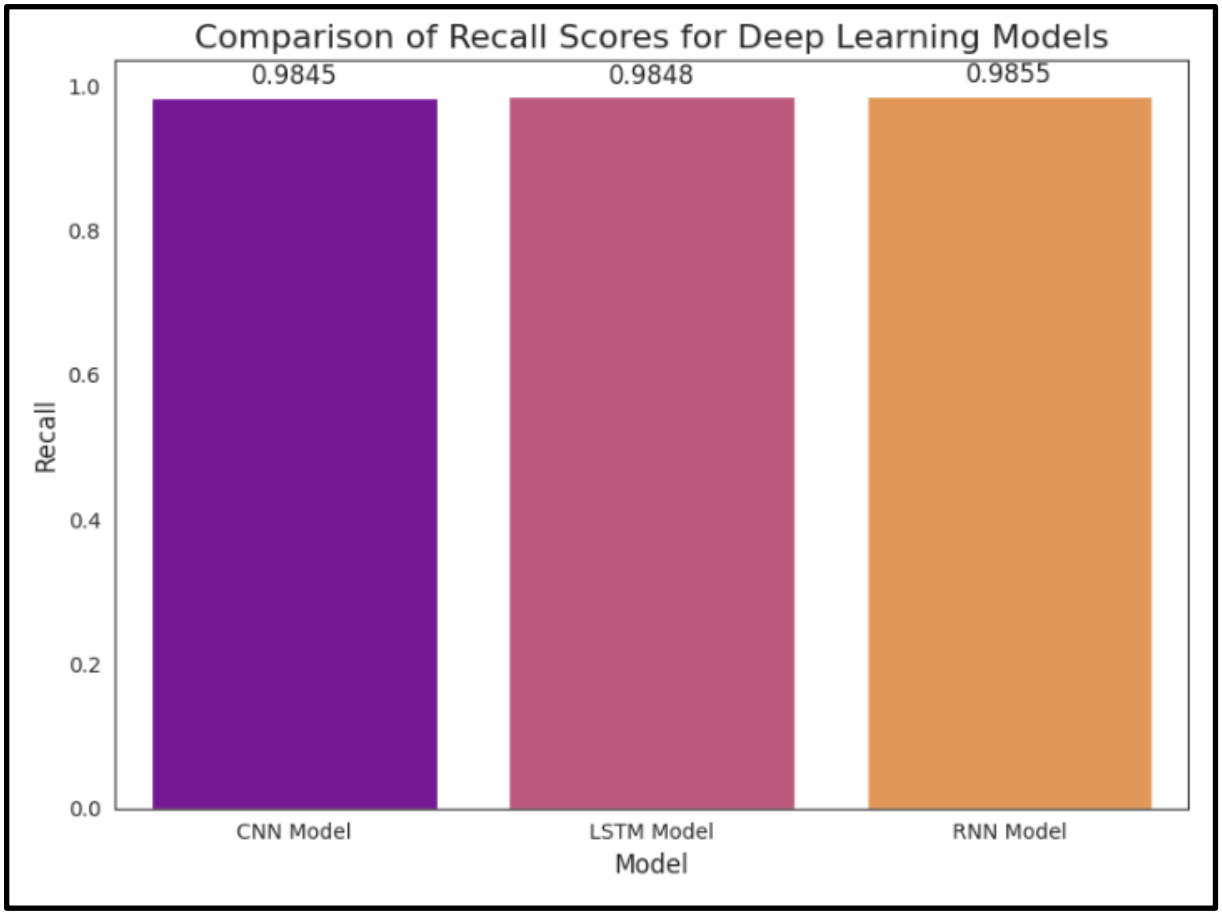


Figure 6.3.9: Bar diagram to compare all models by recall scores

The bar diagram depicts the weighted average recall scores of three models. Nevertheless, the RNN model has the largest recall value of 0.9855 compared to the LSTM and CNN models, indicating that there is a small superiority of RNN in detecting network intrusions.

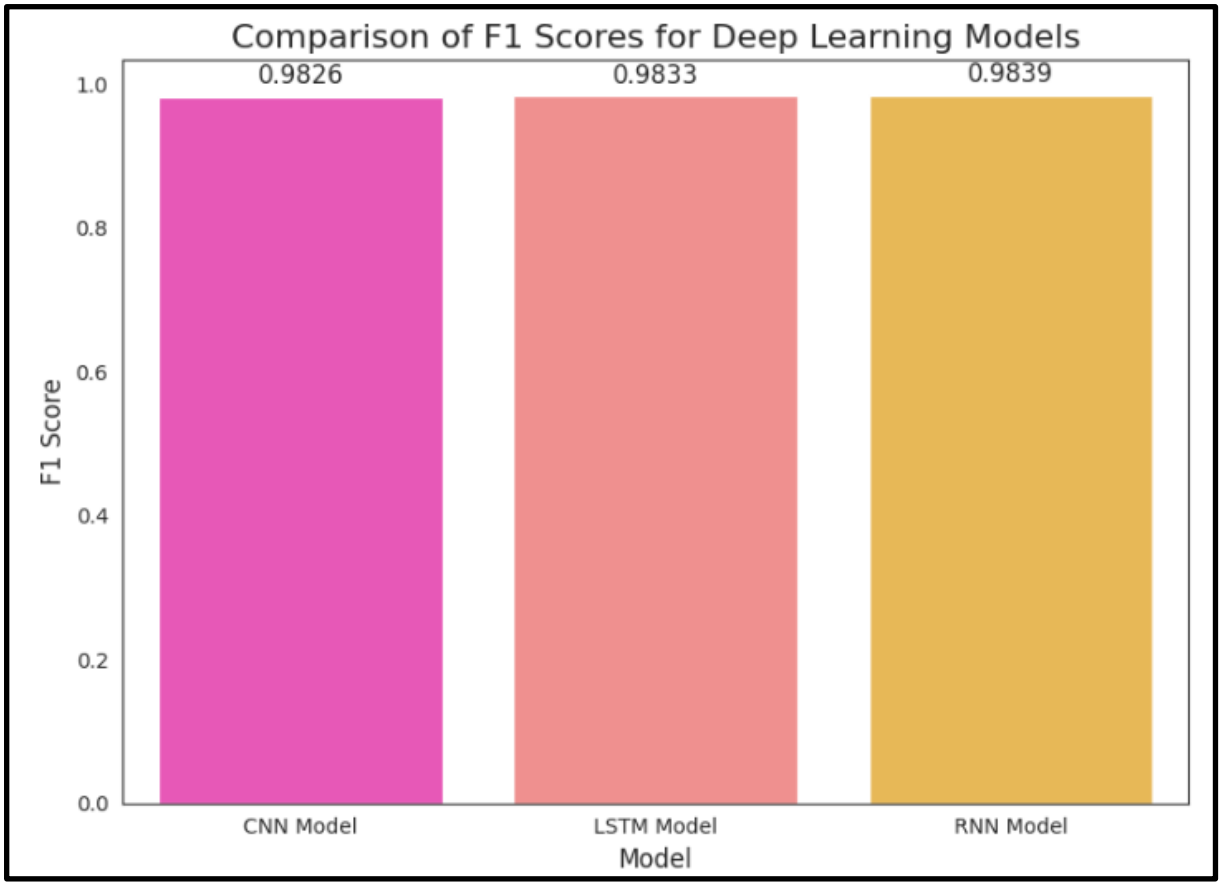


Figure 6.3.10: Bar diagram to compare all models by F1 scores

The bar plot displays that the RNN model has attained the highest weighted average F1 score (0.9839), than the LSTM model (0.9833) and the CNN model (0.9826), demonstrating its excellent predictive capability of network intrusions.

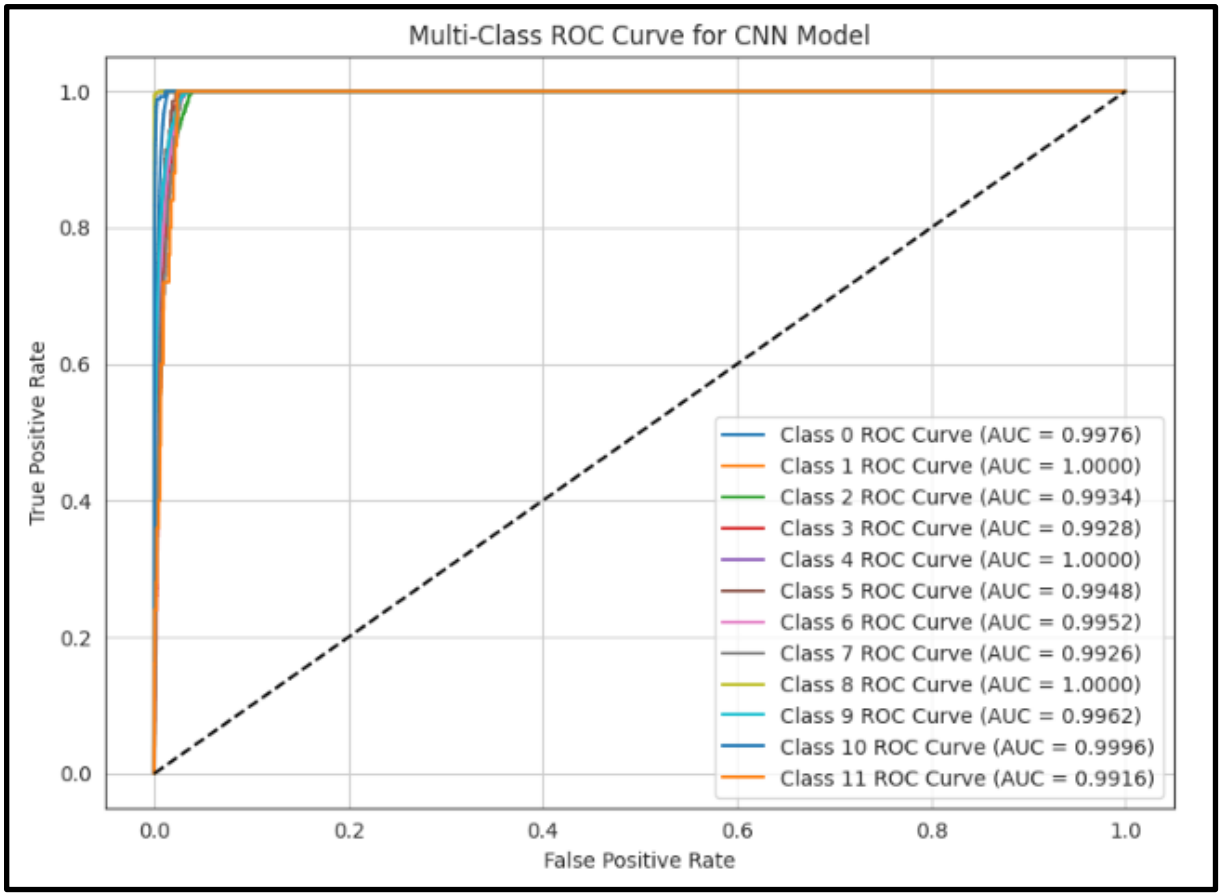


Figure 6.3.11: AUC-ROC curve of CNN

The "AUC-ROC plot" of the CNN model shows that there is an excellent performance exhibited by "Class 1" (AUC = 1.0000), "Class 8" (AUC = 1.0000), and "Class 4" (AUC = 1.0000). These classes were correctly classified with a 100% AUC score. The different classes, such as "Class 2" and "Class 9", also indicated a higher AUC of nearly 1.0, indicating a high effectiveness of the model on multiple types of attacks.

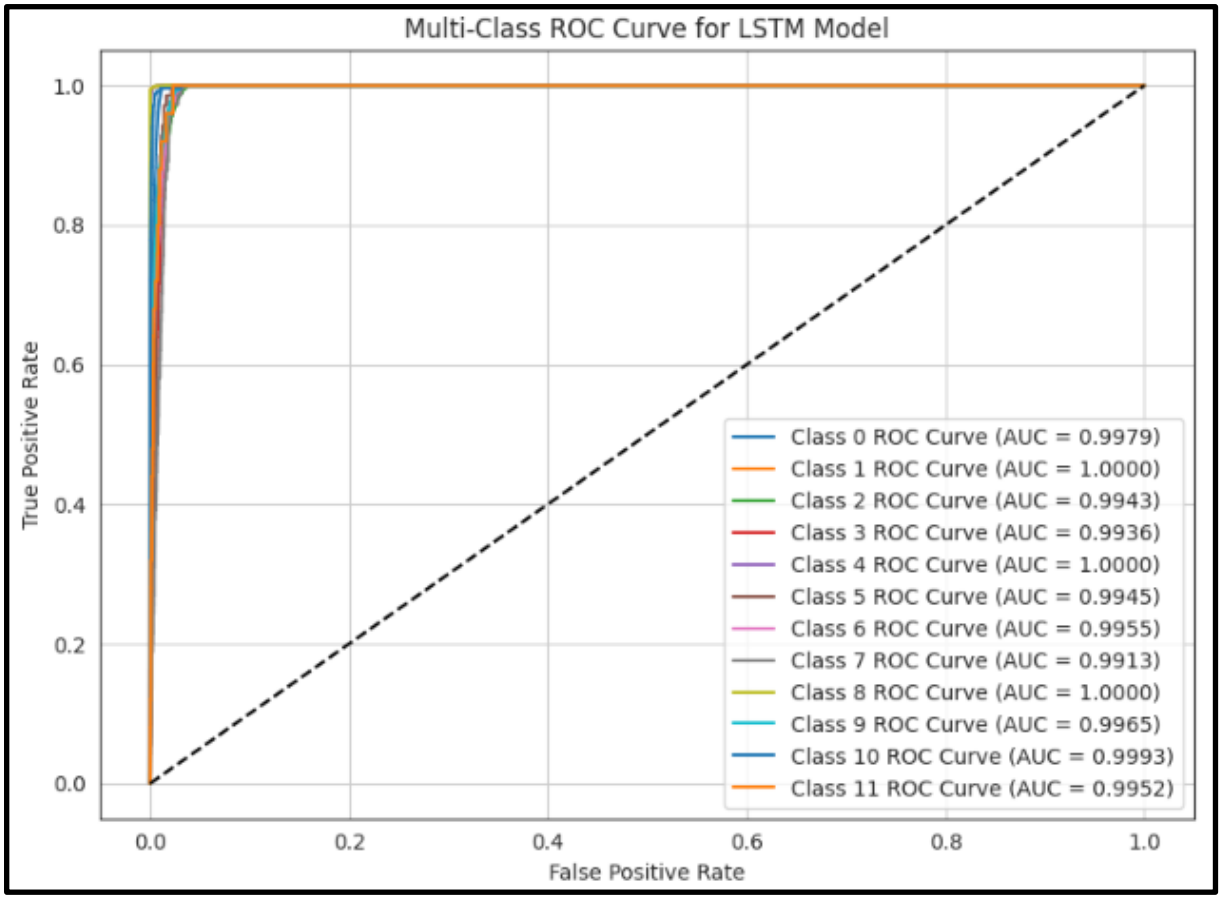


Figure 6.3.12: AUC-ROC curve of LSTM

The "AUC-ROC curve" of the LSTM model shows that Class 1 (BenignTraffic) showed a 100% result with a 1.0 AUC value. Other classes, including Class 8 (Web Application Attack) and Class 4 (DDoS Attack), also showed good results with AUC being above 0.99. Some classes, such as Class 3 (MITM-ArpSpoofing) and Class 11 (DictionaryBruteForce), performed lower AUC scores.

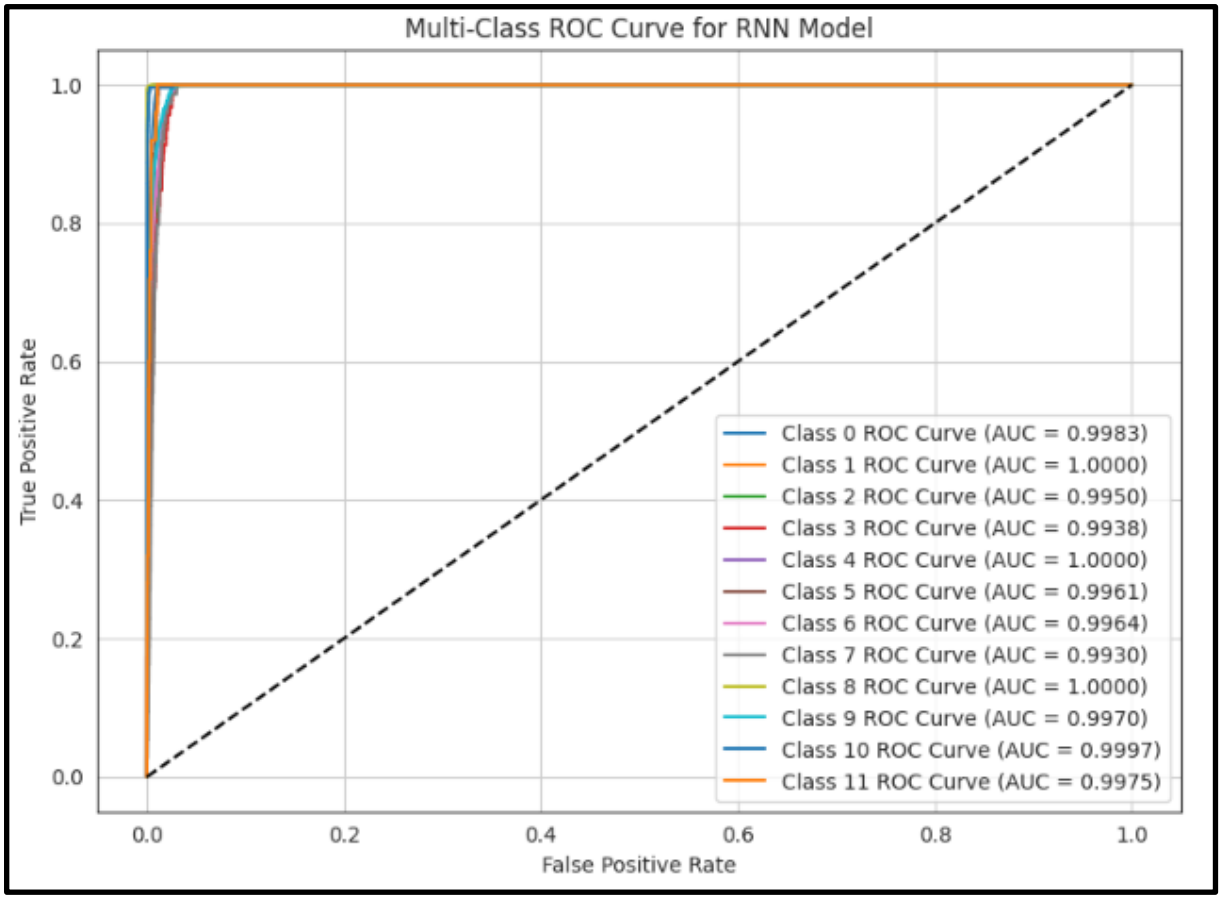


Figure 6.3.13: AUC-ROC curve of RNN

According to the "AUC-ROC curve" of the RNN model, the best performing classes were Class 1 (DoS Attack), Class 8 (Mirai Botnet Attack) and Class 9 (Web Application Attack), whose AUC was close to one. In terms of detection abilities, these classes score very well with high accuracy, which points to the fact that the RNN model successfully estimates these attacks at high levels of perfection.

	Models	Accuracy Score	Precision Score	Recall Score	F1 Score	AUC Score
0	CNN	0.984473	0.984055	0.984473	0.982562	0.996137
1	LSTM	0.984785	0.984103	0.984785	0.983264	0.996515
2	RNN	0.985521	0.985216	0.985521	0.983931	0.997226

Figure 6.3.14: Evaluation metrics of each model

The best results in terms of classifying IoT network intrusions were accomplished by an RNN model, recording an accuracy of 98.55%, precision of 98.52%, recall of 98.55%, F1 score of 98.39%, and AUC of 99.72. This renders it the most conservative model between the CNN and LSTM models in intrusion detection activities.

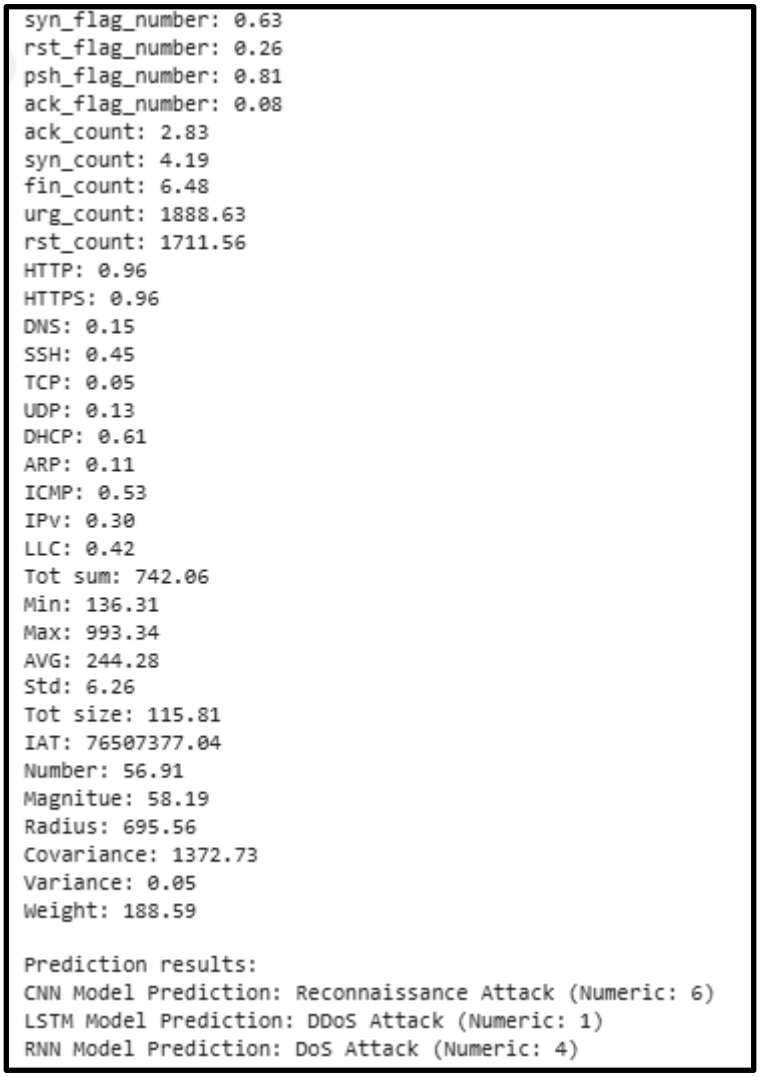


Figure 6.3.15: User interface of IDS with deep learning models

A “user interface” has been designed to test the performance of IDS using three DL models in a text-based user interface. The user interface gave instantaneous predictions of intrusions by using randomly created input data. The user input data is based on the range of collected IoT network data, ensuring realistic intrusion prediction solutions. The CNN model showed the best accuracy as the main prediction was that of DDoS Attack, whereas, LSTM model primarily predicted DDoS Attack with a very high degree of accuracy. The RNN model predicted a mixed outcome, tending to classify it as a DoS Attack, but also at other instances as “BenignTraffic” and “Reconnaissance Attack”.

The system followed through with user inputs, generating real-time intrusion predictions with varying attack labels, thus showing a responsive and efficient intrusion detection interface to be used in various cybersecurity applications.

6.4 Summary

Chapter 6 concludes that the RNN model performs superiorly compared to the CNN and LSTM models. As far as accuracy is concerned, with an astonishing accuracy rate of 98.55%, the RNN model perfectly predicts network intrusions. RNN produced very good prediction scores on DoS, DDoS and Mirai Botnet attacks, having high precision, recall, F1 scores and AUC-ROC curves. Nonetheless, benign traffic was accurately forecasted in all models as a moderate number that constitutes normal traffic in the network. It points out the advantages and disadvantages of every model used to detect different IoT network attacks.

Chapter 7: Evaluation

7.1 Introduction

The evaluation chapter will have a comparison of the current study findings with what has been established in the previous literature in a critical way. It shows the high 'accuracy', 'precision', 'recall', and 'F1 scores' of the RNN model in comparison with the CNN and LSTM, which proves the high performance of the RNN model.

The results are consistent with those of past works, which proves the relevance of "deep learning methods" to improve intrusion tracking on the IoT networks. This comparison confirms the reliability and enhancements of the suggested DL models in the intrusion detection of the network.

7.2 Comparisons of Findings with Existing Literature

The present studies on the improved "intrusion detection system" (IDS) based on deep learning models have demonstrated substantial progress compared to the past works because they measured high scores of accuracies in classifying network intrusions on IoTs. The CNN model of the present study obtained an accuracy of 98.45%, whereas the LSTM and RNN models obtained the accuracies of 98.48% and 98.55%, respectively. These findings demonstrate that deep learning models can produce better results than "traditional machine learning models" in IDS processes since the latter tend to moderate their accuracy in regard to levels achieved in this study. As an example, the "Decision Tree" and "Support Vector Machine" (SVM) machine learning models were identified to reach scores of 93% to 97% with regard to accuracy, and with certain models failing to perform better with specific types of attacks (Araujo, Antunes and Vieira, 2023). The use of CNN, LSTM, and RNN models in the current study has produced better accuracy. It also shows exceptional effectiveness in detecting a wide scope of intrusions, those that are sophisticated, such as DDoS, Mirai Botnet, and SQL injections.

The results of the current study also demonstrated the excellence of deep learning techniques. The proposed RNN model with an accuracy rate of 98.55% beats other "machine learning models", including DNNs. The performance of the RNN model also outperforms hybrid models, like CNN-SVM, that have an accuracy varying between 97% to 99% with occasional failure in meeting the challenge of real-time attack detection (Chua and Salam, 2022). Also, the accuracy, the recall and F1 measures of the "neural network models" surpassed the same done by the machine learning models. For instance, the RNN model achieved an F1 score of 98.39% which was much higher as compared to the F1 score achieved by the machine

learning model, which usually comes in the range of 85% to 95%. Complicated temporal and spatial patterns in an IoT network traffic presented a challenge in traditional models, but the "neural network models" scored highly in this.

The proposed project is able to fill numerous gaps presented within the previous works. A key target of improvement was the depth of the evaluation metrics, mainly precision, recall and F1 scores, which are usually neglected or poorly reported before (Leon et al., 2022). These metrics have been thoroughly computed in the course of the current study and have demonstrated an undeniable superiority over the machine learning modelling techniques. The study not only took into account a high degree of accuracy but also addressed the problem of class imbalance and false positives supported by the previous models (Aljuaid and Alshamrani, 2024). The project exploited methods such as "hybrid deep learning architectures", such as "CNN-LSTM", to enhance the scalability and flexibility of the IDS so that the models could handle new IoT network architectures and attack patterns without incurring performance degradation.

The other critical gap that is covered by the proposed study is the possibility of the models to identify sophisticated and unseen attacks that include zero-day attacks (Bamber et al., 2025). The hybrid deep learning models adapted in this study, specifically, the addition of CNNs to LSTMs, proved better suited to face such cyberattacks and, consequently, the overall effectiveness of the IDS as well. The work has also concentrated on the enhanced detection and responsive measures that will enable instant forecasting and warning based on information on incoming traffic in the network. The application would be a significant advancement to the traditional ML models because they did not stand a chance of complex real-time prediction of cyberattacks (Meliboev et al., 2022).

In conclusion, the current project suggests an algorithmically and data-driven approach to enhancing IDS by applying deep learning solutions and curbing a range of adverse scenarios identified in the prior works and setting new standards in intrusion detection.

7.4 Summary

The comparison of how the present study compares with the past studies has been done in this chapter, and deep learning models fare much better, particularly RNN, in the task of detecting intrusions at IoT networks. The result of this study is reported with high 'accuracy', 'precision', 'recall' and 'F1 scores' of the RNN model compared against traditional machine learning algorithms. The proposed work resolved all gaps that the previous works have in terms of real-time detection, scalability and detection of sophisticated attacks. The study

establishes new standards in the improvement of the performance of “intrusion detection systems” using DL models.

Chapter 8: Conclusions

8.1 Conclusions

The prospect of increasing network cybersecurity through IDS with the help of DL techniques has reached new levels of improving the safety of networks, specifically in IoT implementations. The purpose of the study was effectively fulfilled by training CNN, LSTM and RNN models to classify and determine intrusion with high accuracy. Compared to CNN and LSTM models, the accuracy of RNN was the highest, 98.55% which indicates it is the most reliable model to be used in intrusion detection. The project also covered the literature gaps identified in previous studies, including non-reporting of evaluation metrics and the difficulty of capturing attacks in real-time. The project resolved these concerns with a superior analysis of the performance metrics, such as “precision”, “recall”, “F1-score”, and “AUC-ROC curves”, therefore leading to a better assessment of the performance of IDS models.

The result of the study was compared to existing works, but the result demonstrated better performance. To use an example, the RNN model has been evaluated better than historical “machine learning algorithms” such as the “SVM” and “Decision Trees”, which generally provided accuracy between 93-97 per cent. The precision, recall, and F1 scores of the RNN were significantly better, and, therefore, the model of the RNN was able to manage complicated temporal and spatial patterns of IoT network traffic. Such comparison to various past works demonstrates the effectiveness of the DL technology to handle complex, large-scale data and discriminate against diverse types of attacks, including advanced attacks like “DoS”, “DDoS” and “Mirai Botnet attacks”.

Nevertheless, there are a few limitations in the study. The major issues dealt with were the class imbalance problem that the IoT network traffic dataset presents. Although feature engineering and data preprocessing have been used as a mitigating strategy to mitigate the problem of imbalance, it was still a challenge to the efficacy of the models on minority classes of attacks. Moreover, the models performed well in detecting most of the common intrusions; notwithstanding, those intrusions that are complex or infrequent attacks still had problems in being recognised, especially in real-world settings.

Despite the good results yielded by the deep learning models, some of the attack categories could not be predicted with the same level of precision as others, among them “Web

Applications Attacks", "Malware & Exploit-Based Attacks" and or "Injection Attacks". Further, the models failed to counter high false positive levels of rare categories of intrusions, which implies misclassifications of those cyberattacks or fewer attempts of such attacks in the IoT network environment. In addition, the user interface, due to the text-based interaction, was restrictive, and opportunities were available for further development of a more interactive or visual interface to improve usability. An additional problem was that the IDS was not scalable to larger data sets and variable aggressive patterns. Although the models worked very well on the existing set of data, a larger IoT ecosystem with even more traffic might cause a drop in performance.

The project also tried to apply hyperparameter optimisation and transfer learning as suggested by functional needs, but could not fully do so. Hyperparameter tuning was not necessary because with each model, the accuracy of each was very high, and any further tuning of hyperparameters is not necessary. Transfer learning was not used since the developed neural networks obtained good results without the need to implement the pre-trained models. Thus, in the context of the undertaken analysis, these advanced methods were not so essential in enhancing the performance of the deep learning models.

In order to summarise, the study has been able to fulfil most of its core objectives, although there will be issues relating to intrusion detection, scalability of the models, and explainability of the systems.

8.2 Reflection

The project has helped me learn more about how powerful the use of "deep learning models" can be in improving "intrusion detection systems" (IDS) in IoT networks. I have gained the notion of using "Convolutional Neural Networks" (CNNs), "Long Short-Term Memory" (LSTM) networks, and "Recurrent Neural Networks" (RNNs) to classify network traffic and detect cyberattacks. I have improved my Python programming skills and expertise in terms of the use of 'TensorFlow', 'Keras', and 'scikit-learn', "data preprocessing", "feature engineering", and "DL model evaluation". I have also acquired knowledge of performance measures such as precision, recall, and F1 measures, as well as how to optimise models using them to achieve higher accuracy and efficiency. In future, I would like to work on improving these models that focus on real-time detection and the use of a hybrid architecture of deep learning models to execute more complicated attack patterns in the network traffic of an organisation.

8.3 Further Recommendations

Future improvements in this project should therefore consider issues that will address the limitations in areas like class imbalance, rare cyberattack prediction and system scalability. The models should also be improved to work with imbalanced data, probably by oversampling or more comprehensive methods of feature engineering (Madhu et al., 2023). In addition, the detection of the rare or complex attacks must be enhanced, particularly in realistic settings, where they have the potential to result in a very serious threat. It should also be used in future studies to increase the scalability of a system, whereby the models can handle large amounts of IoT data, in such a way that traffic patterns can be accommodated without compromising the accuracy of the model.

Additionally, a more friendly interface is required such as a graphical user interface (GUI) which will be used to replace the text-based interface since it will be more interactive and easier to interface with (Chaganti et al., 2023). The application of the transfer learning and hyperparameter optimisation methods to even enhance the potential of the models and curb overfitting and underfitting should be considered in the study as well as possible.

Finally, it is important that the deep learning models are made transparent enough to promote belief and acquaintance with the models, which should be prioritised in future developments, as well. This would ensure robustness and scalability of intrusion detection systems and be better prepared to combat more cybersecurity problems.

Reference list

Abdalzaher, M., Seddik, K., Elsabrouty, M., Muta, O., Furukawa, H. and Abdel-Rahman, A. (2016). Game Theory Meets Wireless Sensor Networks Security Requirements and Threats Mitigation: A Survey. *Sensors*, [online] 16(7), p.1003. doi:https://doi.org/10.3390/s16071003.

Ajagbe, S.A., Awotunde, J.B. and Florez, H. (2024). Intrusion Detection: A Comparison Study of Machine Learning Models Using Unbalanced Dataset. *SN Computer Science*, [online] 5(8), pp.1–10. doi:https://doi.org/10.1007/s42979-024-03369-0.

Ali, M.L., Thakur, K., Schmeelk, S., Debello, J. and Dragos, D. (2025). Deep Learning vs. Machine Learning for Intrusion Detection in Computer Networks: A Comparative Study. *Applied Sciences*, [online] 15(4), p.1903. doi:https://doi.org/10.3390/app15041903.

Al-Imran, M. and Ripon, S.H. (2021). Network Intrusion Detection: An Analytical Assessment Using Deep Learning and State-of-the-Art Machine Learning Models. *International Journal of Computational Intelligence Systems*, [online] 14(1), pp.1–20. doi:https://doi.org/10.1007/s44196-021-00047-4.

Aljuaid, W.H. and Alshamrani, S.S. (2024). A Deep Learning Approach for Intrusion Detection Systems in Cloud Computing Environments. *Applied Sciences*, [online] 14(13), p.5381. doi:https://doi.org/10.3390/app14135381.

Almalki, S.S. (2025). A Deep Learning-Based Framework for Real-Time Detection of Cybersecurity Threats in IoT Environments. *International Journal of Advanced Computer Science and Applications*, [online] 16(3), pp.1–10. doi:https://doi.org/10.14569/ijacsa.2025.0160343.

Almohaimeed, M., Alyoubi, R., Aljohani, A., Alhaidari, M., Albalwy, F., Ghabban, F., Alfadli, I. and Ameerbakhsh, O. (2025). Use of Machine Learning and Deep Learning in Intrusion Detection for IoT. *Advances in Internet of Things*, [online] 15(02), pp.17–32. doi:https://doi.org/10.4236/ait.2025.152002.

Alsakran, F., Bendiab, G., Shiaeles, S. and Kolokotronis, N. (2020). *Intrusion Detection Systems for Smart Home IoT Devices: Experimental Comparison Study*. [online] arXiv.org. Available at: https://arxiv.org/abs/2101.06519 [Accessed 2025].

Alsubaei, F.S. (2025). Smart deep learning model for enhanced IoT intrusion detection. *Scientific Reports*, [online] 15(1), pp.1–23. doi:https://doi.org/10.1038/s41598-025-06363-5.

Araujo, I., Antunes, N. and Vieira, M. (2023). Evaluation of Machine Learning for Intrusion Detection in Microservice Applications. *LADC '23: Proceedings of the 12th Latin-American Symposium on Dependable and Secure Computing*, [online] pp.126–135. doi:https://doi.org/10.1145/3615366.3615375.

Asgharzadeh, H., Ghaffari, A., Masdari, M. and Gharehchopogh, F.S. (2024). An Intrusion Detection System on The Internet of Things Using Deep Learning and Multi-objective Enhanced Gorilla Troops Optimizer. *Journal of Bionic Engineering*, [online] 21(5), pp.2658–2684. doi:https://doi.org/10.1007/s42235-024-00575-7.

Awajan, A. (2023). A Novel Deep Learning-Based Intrusion Detection System for IoT Networks. *Computers*, [online] 12(2), p.34. doi:https://doi.org/10.3390/computers12020034.

Bamber, S.S., Katkuri, A.V.R., Sharma, S. and Angurala, M. (2025). A hybrid CNN-LSTM approach for intelligent cyber intrusion detection system. *Computers & Security*, [online] 148, p.104146. doi:https://doi.org/10.1016/j.cose.2024.104146.

Bazen, A., Barg, F.K. and Takeshita, J. (2021). Research techniques made simple: An introduction to qualitative research. *Journal of Investigative Dermatology*, [online] 141(2), pp.241–247. doi:https://doi.org/10.1016/j.jid.2020.11.029.

Celik, Y., Basaran, E. and Goel, S. (2025). Deep Learning Methods for Intrusion Detection Systems on the CSE-CIC-IDS2018 Dataset: A Review. *Digital Forensics and Cyber Crime* , [online] pp.38–65. doi:https://doi.org/10.1007/978-3-031-89363-6_3.

Chaganti, R., Suliman, W., Ravi, V. and Dua, A. (2023). Deep Learning Approach for SDN-Enabled Intrusion Detection System in IoT Networks. *Information*, [online] 14(1), p.41. doi:https://doi.org/10.3390/info14010041.

Cheng, Y., Fu, H. and Sun, X. (2021). Intrusion Detection Based on the Game Theory. *International Conference on Frontiers of Electronics, Information and Computation Technologies*, [online] pp.1–7. doi:https://doi.org/10.1145/3474198.3478267.

Chua, T.-H. and Salam, I. (2022). Evaluation of Machine Learning Algorithms in Network-Based Intrusion Detection System. *arXiv (Cornell University)*, [online] pp.1–31. doi:https://doi.org/10.48550/arxiv.2203.05232.

Chua, T.-H. and Salam, I. (2023). Evaluation of Machine Learning Algorithms in Network-Based Intrusion Detection Using Progressive Dataset. *Symmetry*, [online] 15(6), p.1251. doi:https://doi.org/10.3390/sym15061251.

Durgaraju, S. and Vel, D.V.T. (2025). A Systematic Review of Deep Learning Models for Intrusion Detection: From CNN to Hybrid Architectures. *Communications in Computer and Information Science*, [online] pp.50–63. doi:https://doi.org/10.1007/978-3-031-92041-7_5.

Elnakib, O., Shaaban, E., Mahmoud, M. and Emara, K. (2023). EIDM: deep learning model for IoT intrusion detection systems. *The Journal of Supercomputing*, [online] 79, pp.13241–13261. doi:https://doi.org/10.1007/s11227-023-05197-0.

Elrawy, M.F., Awad, A.I. and Hamed, H.F.A. (2018). Intrusion detection systems for IoT-based smart environments: a survey. *Journal of Cloud Computing*, [online] 7(1), pp.1–20. doi:https://doi.org/10.1186/s13677-018-0123-6.

Faker, O. and Dogdu, E. (2019). Intrusion Detection Using Big Data and Deep Learning Techniques. *Proceedings of the 2019 ACM Southeast Conference on ZZZ - ACM SE '19*, [online] pp.86–93. doi:https://doi.org/10.1145/3299815.3314439.

Farhan, M., Waheed ud din, H., Ullah, S., Hussain, M.S., Khan, M.A., Mazhar, T., Khattak, U.F. and Jaghdam, I.H. (2025). Network-based intrusion detection using deep learning technique. *Scientific Reports*, [online] 15(1), pp.1–25. doi:https://doi.org/10.1038/s41598-025-08770-0.

Fortinet (2025). *What is a Cyber Attack? Definition & Prevention*. [online] www.fortinet.com. Available at: https://www.fortinet.com/resources/cyberglossary/what-is-cyber-attack [Accessed 2025].

Harshavardhanan, P., Muneeswaran, V., Saravanan, D., Vijayakarthik, P., Natchadalingam, R. and Shaker, C. (2023). An Adaptive Intrusion Detection System for IoT Networks Using Deep Learning. *2023 IEEE Fifth International Conference on Advances in Electronics,*

Computers and Communications (ICAECC), [online] pp.1–6. doi:https://doi.org/10.1109/icaecc59324.2023.10560150.

Hindy, H., Atkinson, R., Tachtatzis, C., Colin, J.-N., Bayne, E. and Bellekens, X. (2020). Utilising Deep Learning Techniques for Effective Zero-Day Attack Detection. *Electronics*, [online] 9(10), p.1684. doi:https://doi.org/10.3390/electronics9101684.

Hnamte, V., Nhung-Nguyen, H., Hussain, J. and Hwa-Kim, Y. (2023). A Novel Two-Stage Deep Learning Model for Network Intrusion Detection: LSTM-AE. *IEEE Access*, [online] 11, pp.37131–37148. doi:https://doi.org/10.1109/ACCESS.2023.3266979.

Islam, M., Ayan, J., Kais, E., Das, R.K. and Islam, M.M. (2024). Evaluating Deep Learning Models for Network Intrusion Detection: A Comparative Analysis. *Conference: 2024 27th International Conference on Computer and Information Technology (ICCIT)*, [online] pp.3635–3640. doi:https://doi.org/10.1109/iccit64611.2024.11022010.

Isong, B., Kgote, O. and Abu-Mahfouz, A. (2024). Insights into Modern Intrusion Detection Strategies for Internet of Things Ecosystems. *Electronics*, [online] 13(12), p.2370. doi:https://doi.org/10.3390/electronics13122370.

Jablaoui, R. and Liouane, N. (2025). Network security based combined CNN-RNN models for IoT intrusion detection system. *Peer-to-Peer Networking and Applications*, [online] 18(3), pp.1–25. doi:https://doi.org/10.1007/s12083-025-01944-7.

Kamalakkannan, D., Menaga, D., Shobana, S., Daya, V., Rajagopal, R. and Tiwari, M. (2023). A Detection of Intrusions Based on Deep Learning. *Cybernetics and Systems*, [online] 56(5), pp.1–15. doi:https://doi.org/10.1080/01969722.2023.2175134.

Kamil, W.F. and Mohammed, I.J. (2023). Deep learning model for intrusion detection system utilizing convolution neural network. *Open Engineering*, [online] 13(1), pp.1–11. doi:https://doi.org/10.1515/eng-2022-0403.

Khan, A.R., Kashif, M., Jhaveri, R.H., Raut, R., Saba, T. and Bahaj, S.A. (2022). Deep Learning for Intrusion Detection and Security of Internet of Things (IoT): Current Analysis, Challenges, and Possible Solutions. *Security and Communication Networks*, [online] 2022, pp.1–13. doi:https://doi.org/10.1155/2022/4016073.

Leon, M., Markovic, T. and Punnekkat, S. (2022). Comparative Evaluation of Machine Learning Algorithms for Network Intrusion Detection and Attack Classification. *2022 International Joint Conference on Neural Networks (IJCNN)*, [online] pp.1–17. doi:https://doi.org/10.1109/ijcnn55064.2022.9892293.

Madhu, B., Venu Gopala Chari, M., Vankdothu, R., Silivery, A.K. and Aerranagula, V. (2023). Intrusion detection models for IOT networks via deep learning approaches. *Measurement: Sensors*, [online] 25, p.100641. doi:https://doi.org/10.1016/j.measen.2022.100641.

Maseno, E.M., Wang, Z. and Sun, Y. (2024). Performance Evaluation of Intrusion Detection Systems on the TON_IoT Datasets Using a Feature Selection Method. *Proceedings of the 2024 8th International Conference on Computer Science and Artificial Intelligence*, [online] pp.607–613. doi:https://doi.org/10.1145/3709026.3709048.

Mejdi, H., Elmadssia, S., Koubãa, M. and Ezzedine, T. (2024). A Comprehensive Survey on Game Theory Applications in Cyber-Physical System Security: Attack Models, Security Analyses, and Machine Learning Classifications. *IEEE Access*, [online] 12, pp.163638–163653. doi:https://doi.org/10.1109/access.2024.3491502.

Meliboev, A., Alikhanov, J. and Kim, W. (2022). Performance Evaluation of Deep Learning Based Network Intrusion Detection System across Multiple Balanced and Imbalanced Datasets. *Electronics*, [online] 11(4), p.515. doi:https://doi.org/10.3390/electronics11040515.

Mirsky, Y., Doitshman, T., Elovici, Y. and Shabtai, A. (2018). Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. *arXiv:1802.09089 [cs]*, [online] pp.1–10. doi:https://doi.org/10.48550/arXiv.1802.09089.

Mohale, V.Z. and Obagbuwa, I.C. (2025). Evaluating machine learning-based intrusion detection systems with explainable AI: enhancing transparency and interpretability. *Frontiers in Computer Science*, [online] 7, pp.1–17. doi:https://doi.org/10.3389/fcomp.2025.1520741.

Noori Shaker, B., Qasim Al-Musawi, B. and Falih Hassan, M. (2023). A Comparative Study of IDS-Based Deep Learning Models for IoT Network. *In 2023 International Conference on Advances in Artificial Intelligence and Applications AAIA 2023)*, [online] pp.1–10. doi:https://doi.org/10.1145/3603273.3635058.

Saranya, T., Sridevi, S., Deisy, C., Chung, T.D. and Khan, M.K.A.Ahamed. (2020). Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review. *Procedia Computer Science*, [online] 171, pp.1251–1260. doi:https://doi.org/10.1016/j.procs.2020.04.133.

Shaker, B.N., Al-Musawi, B.Q. and Hassan, M.F. (2023). A Comparative Study of IDS-Based Deep Learning Models for IoT Network. *AAIA 2023: 2023 International Conference on Advances in Artificial Intelligence and Applications*, [online] pp.18–20. doi:https://doi.org/10.1145/3603273.3635058.

Subba, B., Biswas, S. and Karmakar, S. (2018). A game theory based multi layered intrusion detection framework for VANET. *Future Generation Computer Systems*, [online] 82, pp.12–28. doi:https://doi.org/10.1016/j.future.2017.12.008.

Talukder, Md.A., Islam, Md.M., Uddin, M.A., Hasan, K.F., Sharmin, S., Alyami, S.A. and Moni, M.A. (2024). Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction. *Journal of Big Data*, [online] 11(1), pp.1–20. doi:https://doi.org/10.1186/s40537-024-00886-w.

Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S.A.R., Ghogho, M. and El Moussa, F. (2020). DeepIDS: Deep Learning Approach for Intrusion Detection in Software Defined Networking. *Electronics*, [online] 9(9), p.1533. doi:https://doi.org/10.3390/electronics9091533.

Tsimenidis, S., Lagkas, T. and Rantos, K. (2021). Deep Learning in IoT Intrusion Detection. *Journal of Network and Systems Management*, [online] 30(1), pp.1–20. doi:https://doi.org/10.1007/s10922-021-09621-9.

Vishwakarma, M. and Kesswani, N. (2022). DIDS:A deep neural network based real-time intrusion detection system for IoT. *Decision Analytics Journal*, [online] 5, p.100142. doi:https://doi.org/10.1016/j.dajour.2022.100142.

Wang, Z., Xu, S., Xu, G., Yin, Y., Zhang, M. and Sun, D. (2020). Game Theoretical Method for Anomaly-Based Intrusion Detection. *Security and Communication Networks*, [online] 2020, p.e8824163. doi:https://doi.org/10.1155/2020/8824163.

Zhang, Y., Muniyandi, R.C. and Qamar, F. (2025). A Review of Deep Learning Applications in Intrusion Detection Systems: Overcoming Challenges in Spatiotemporal Feature Extraction and Data Imbalance. *Applied Sciences*, [online] 15(3), p.1552. doi:https://doi.org/10.3390/app15031552.