EXPERIMENTS OF RECURRENT NEURAL NETWORK MODELS FOR INTRUSION DETECTION ON NSL-KDD DATASET

TEO WEI CHEN

UNIVERSITI KEBANGSAAN MALAYSIA

EXPERIMENTS OF RECURRENT NEURAL NETWORK MODELS FOR INTRUSION DETECTION ON NSL-KDD DATASET

TEO WEI CHEN

PROJECT SUBMITTED IN PARTIAL FULFILMENT FOR THE DEGREE OF MASTER OF CYBER SECURITY

FACULTY OF INFORMATION SCIENCE AND TECHNOLOGY
UNIVERSITI KEBANGSAAN MALAYSIA
BANGI

2024

EXPERIMENTS OF RECURRENT NEURAL NETWORK MODELS FOR INTRUSION DETECTION ON NSL-KDD DATASET

TEO WEI CHEN

PROJEK YANG DIKEMUKAKAN UNTUK MEMENUHI SEBAHAGIAN DARIPADA SYARAT IJAZAH SARJANA KESELAMATAN SIBER

FAKULTI TEKNOLOGI DAN SAINS MAKLUMAT
UNIVERSITI KEBANGSAAN MALAYSIA
BANGI

2024

DECLARATION

I hereby declare that the work in this thesis is my own except for quotations and summaries which have been duly acknowledged.

19 February 2024

TEO WEI CHEN
P117987

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my project supervisor Mohd Zamri Murah and my examiner Dr. Wan Fariza for their continuous patience and motivational support. Their guidance and knowledge input has helped me in successfully completing this project within the recommended timeframe. Moreover, I would also like to extend my thanks to the Program Coordinator, all the lecturers of Master of Cybersecurity and staff from the Faculty of Information Science & Technology, for their constant support and encouragement during my academic journey.

Furthermore, I would like to express my heartfelt gratitude to my parents, brothers, and all the course mates, thanks for their supports and encouragement throughout my academic journey.

Lastly, I express my sincere thanks to all those who have contributed to this work in one way or another. My hope is that this study will, to some extent, contribute to the field of research and serve as a reference for future researchers.

ABSTRAK

Dalam masyarakat moden hari ini, bilangan serangan siber berkembang pesat, dan kita menghadapi serangan pencerobohan rangkaian moden dan kompleks setiap hari, walaupun dalam rangkaian komputer yang selamat. Walau bagaimanapun, sistem pengesanan pencerobohan tradisional berdasarkan pemilihan dan klasifikasi ciri mempunyai beberapa kelemahan, seperti memproses maklumat berlebihan dan meningkatkan masa pengiraan. Untuk mengelakkan pelanggaran, sangat diperlukan bagi pentadbir keselamatan untuk mengesan penceroboh dan menghalangnya memasuki rangkaian. Teknik pembelajaran mesin digunakan untuk menyelesaikan jenis masalah ini, tetapi mereka tidak dapat menggeneralisasi kerana mereka gagal mendapatkan hubungan antara ciri-ciri. Dalam kertas ini, kami membincangkan bagaimana melakukan eksperimen menggunakan seni bina rangkaian saraf berulang RNN (Simple-RNN, LSTM, dan GRU), misalnya, menggunakan pemprosesan data, klasifikasi serangan, pelbagai jenis parameter, dan saiz kumpulan. Kami membandingkan setiap seni bina RNN pada set data pencerobohan rangkaian (NSL-KDD) dan menyimpulkan dengan masa dan hasil ketepatan yang berbeza. Pada akhirnya, IDS berasaskan RNN yang dicadangkan dalam kertas ini, diarkibkan dengan klasifikasi binari untuk RNN-IDS dengan 64 unit, mempunyai ketepatan 79.33%, dan multi-klasifikasi untuk LSTM-IDS dengan 256-unit mempunyai ketepatan 69.50%. Hasil penyelidikan dalam kertas ini bertujuan untuk memilih model RNN yang paling sesuai untuk setiap klasifikasi, kaedah prapemprosesan data dan penalaan hiperparameter untuk sistem pengesanan pencerobohan pembelajaran mendalam.

ABSTRACT

In modern society today, the rapid surge in cyberattacks presents an escalating challenge, with sophisticated network intrusion attempts occurring regularly even within ostensibly secure computer networks. Despite the prevalence of traditional intrusion detection systems relying on feature selection and classification, they grapple with certain limitations, notably the processing of redundant information leading to increased computational time. The urgency for security administrators to detect and thwart intruders from breaching network defences is undeniable. While machine learning techniques have been employed to address these issues, their efficacy is hindered by a limited ability to generalize, often failing to discern intricate relationships among features. In this paper, we discussed how to do an experiment using RNN recurrent neural network architectures (Simple-RNN, LSTM, and GRU), for example, using data preprocessing, classification of the attacks, different types of parameters, and batch size. We compared each of the RNN architectures on network intrusion datasets (NSL-KDD) and concluded with different time and accuracy results. At the end, the RNN-based IDS proposed in this paper, archived with binary classification for RNN-IDS with 64 units, had an accuracy of 79.33%, and multi-classification for LSTM-IDS with 256 units had an accuracy of 69.50%. The research results in this paper aim to choose the most suitable RNN models for each of the classification, data preprocessing methods, and hyperparameter tuning for deep learning intrusion detection systems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

Pictures No. **Page**

LIST OF ABBREVIATIONS

AE	Auto Encoder
AI	Artificial Intelligence
BLSTM	Bidirectional Long Short-Term Memory
CNN	Convolution Neural Network
DDoS	Distributed Denial of Service
DL	Deep Learning
DNN	Deep Neural Network
DoS	Denial of Service
FNN	Feedforward Neural Network
FN	False Negative
FP	False Positive
GRU	Gated Recurrent Unit
HIDS	Host-based Intrusion Detection System
IDS	Intrusion Detection System
LC	Logistic Regression
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multi-Layer Perceptron
NIDS	Network-based Intrusion Detection System
RF	Random Forest
RNN	Recurrent Neural Network
R2L	Remote-to-Local
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
U2R	Synthetic Minority Oversampling

CHAPTER I

INTRODUCTION

1.1 INTRODUCTION

Nowadays, the internet has become the most common technology capable of meeting every human need. Dialup, broadband, Ethernet, and wireless technologies are used to link all PCs and smart gadgets to the internet. Unfortunately, because the internet is the primary link connecting all these smart gadgets, it may expose people to a broad range of potential hazards. An illegal action on a network technology is referred to as a network intrusion. Network intrusion sometimes include the theft of valuable network resources and virtually always compromise network or data security.

Computer networks becoming more essential because of growing usage in various areas and applications. Businesses adopt a defensive stance against network attacks, employing established security tools like firewalls, anti-spam protocols, and antivirus software. Nonetheless, these protective measures reveal a limitation, they struggle to detect emerging or intricate threats. Consequently, Network Intrusion Detection Systems (NIDS) have evolved into a secondary defence layer, actively overseeing network traffic to flag any potential intrusions. This technology serves as a highly efficient defensive mechanism, adept at identifying and mitigating complex cyber threats and attacks (Hnamte et al., 2023).

NIDS serves as a valuable tool in identifying an array of network threats, encompassing distributed denial-of-service (DDoS) attacks, worms, and viruses. Its effectiveness relies heavily on the triumvirate of reliability, precision, and swift detection capabilities. A significant amount of research has been dedicated to Network Intrusion Detection Systems (NIDS), but there's still room for improvement, especially in reducing false alarms and boosting detection accuracy. To tackle these issues, various

machine learning (ML) techniques are being utilized within NIDS to minimize false positives and enhance the system's ability to detect threats accurately. Among these approaches, deep learning (DL) stands out as an advanced methodology contributing to the evolution of NIDS capabilities (Ahmed et al., 2022).

In this project, we do an investigation into the cutting-edge realm of intrusion detection by harnessing the power of deep learning models, specifically in Deep Neural Network (DNN), Convolutional Neural Network (CNN), Recurrent Neural Networks (RNN) models, which are Simple-RNN, Long Short-Term Memory (LSTM) networks and Gated Recurrent Unit (GRU). Deep learning, a subfield of machine learning, has demonstrated remarkable capabilities in extracting complex patterns and features from data, making it a promising avenue for enhancing intrusion detection capabilities. The choice of RNN models for intrusion detection is driven by their ability to capture temporal dependencies, learn complex patterns, handle variable-length inputs, and effectively model contextual information inherent in network traffic data. The NSL-KDD dataset from the Canadian Institute for Cybersecurity (the updated version of the original KDD Cup 1999 Data (KDD99) is used in this project.

At the end of this project, we done the experiments by evaluate and compare the performance for each of the RNN models when applied to intrusion detection in contrast to traditional IDS methods. This performance evaluation encompasses several key aspects, including accuracy, efficiency, adaptability, and scalability. By investigating the strengths and weaknesses of all the RNN models in the context of intrusion detection, we aim to provide insights that can inform the development of more robust and resilient security measures for the digital landscape.

1.2 RESEARCH BACKGROUND

1.2.1 Intrusion Detection System (IDS)

In the 1980s, cybersecurity saw a game-changing moment with the emergence of intrusion detection systems (IDS). These initial systems were quite straightforward, relying on predefined rules to signal any suspicious activities already on their radar. James P. Anderson, a trailblazer in computer security, introduced this concept through a report he authored for the U.S. Air Force in 1980. Anderson's pioneering work laid the essential groundwork for the earliest IDS models, which primarily sifted through system logs to spot potential security breaches (Frąckiewicz, 2023).

An intrusion detection system (IDS) acts as a vigilant overseer, continuously monitoring network traffic to identify any unusual or suspicious activities, promptly alerting the administrator upon detection. The primary function of an IDS lies in the detection of anomalous activities and subsequent reporting to the network administrator. Initially implemented alongside firewalls, IDS proves to be a potent technique capable of effectively managing various security attacks and identifying abnormal behaviours within a target application or computer. Within the realm of IDS systems, two distinctive methodologies play a prominent role: signature-based (misuse) detection and anomaly-based detection. The signature-based approach focuses on pinpointing known attacks through methods such as pattern matching or rule-centric techniques. Conversely, anomaly detection takes a broader approach, aiming to uncover both familiar and unfamiliar attacks by closely analysing their behavioural characteristics. This dual approach allows IDS to provide a robust defence against a spectrum of potential security threats. (Althubiti et al., 2018).

Misuse detection, reliant on attack signatures, operates through multi-class classification, targeting known malicious behaviours. Yet, its drawback lies in missing new attacks without recognizable signatures in the IDS database. Still, these systems excel in accurately pinpointing known malicious behaviours and their variations. On the flip side, anomaly detection-based IDS methods shine in uncovering new attacks by analysing users' typical behaviour profiles. However, their classification is limited to binary, distinguishing solely between normal and anomalous behaviour. This method relies on deviations from norms, allowing for the discovery of previously unseen threats

while categorizing behaviour in only two distinct groups. (Lansky et al., 2021). Figure 1.1 shows the design flow for IDS (Swanagan, 2019).

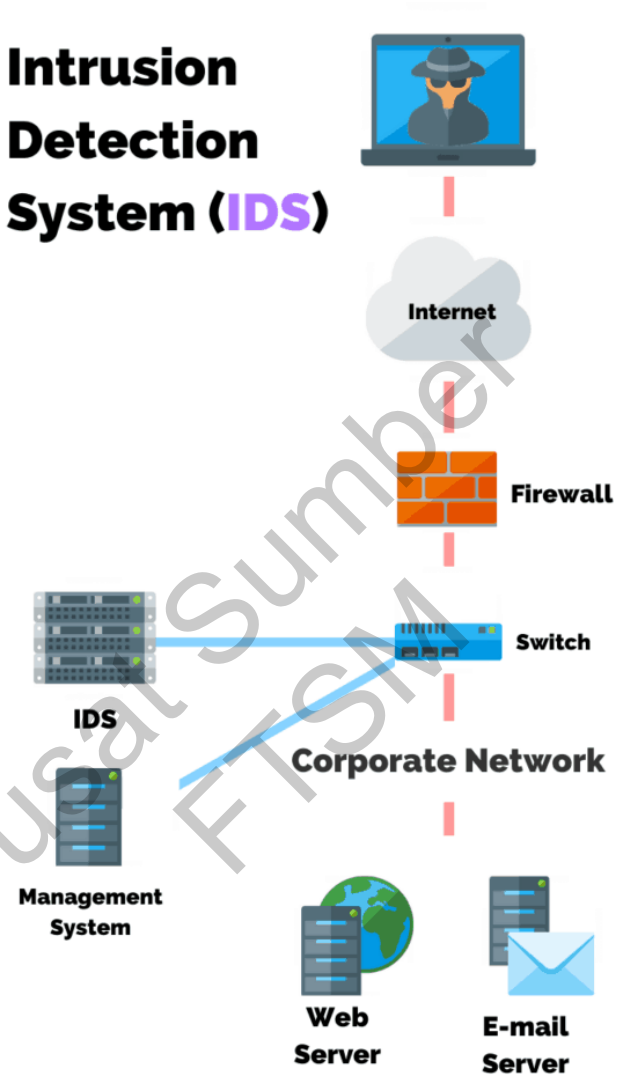


Figure 1.1 Design flow for IDS (Swanagan, 2019)

Numerous methods exist for promptly identifying potentially harmful network activities, aiming to curtail or avert potential damage. Intrusion Detection Systems (IDSs) stand out as pivotal network security technologies, actively scrutinizing and analysing network traffic to pinpoint irregularities or recognize patterns (fingerprints) indicative of previously encountered intrusive activities. Within the realm of IDS, two

primary categories emerge: the Network Intrusion Detection System (NIDS) and the Host-Based Intrusion Detection System (HIDS).

1. Network-based Intrusion Detection Systems (NIDSs) play a crucial role in tracking and assessing all network traffic flowing to and from individual devices. These systems are strategically placed within the network infrastructure to analyze the traffic originating from all connected devices. By observing the entirety of passing traffic within the subnet, NIDS compares this data against a database of known attack signatures. When an attack is identified or unusual behavior is detected, the system generates an alert to notify the administrator. For instance, situating a NIDS on the subnet containing firewalls allows it to scrutinize attempts aimed at breaching the firewall's defenses. In the figure 1.2 below shows the network-based intrusion detection system (GeeksForGeeks, 2019).

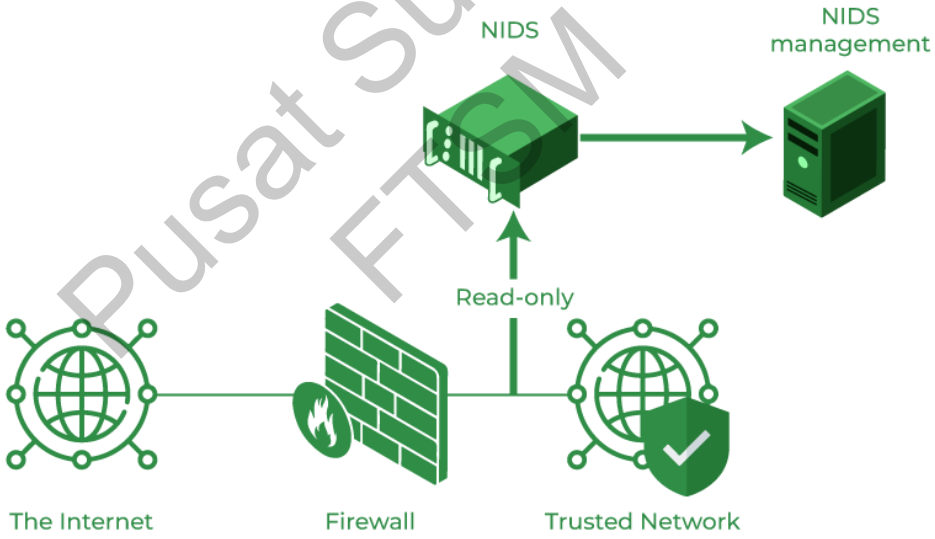


Figure 1.2 Network-based Intrusion Detection System (GeeksForGeeks, 2019)

2. Host-based intrusion detection systems (HIDSs) focus on tracking and analyzing the internal components of an operating system, including crucial system files. Deployed on individual hosts or devices within the network, HIDSs monitor the inbound and outbound packets specific to the device, promptly alerting administrators upon detecting any suspicious or malicious activities. One of the key functionalities involves creating snapshots of the system's existing files and comparing them against previous snapshots. Any alterations or deletions in these critical system files trigger an alert, prompting further investigation by the administrator. Notably, HIDSs find application in safeguarding mission-critical machines that are anticipated to maintain a consistent layout without substantial changes. In the figure 1.3 below shows the host-based intrusion detection system (GeeksForGeeks, 2019).

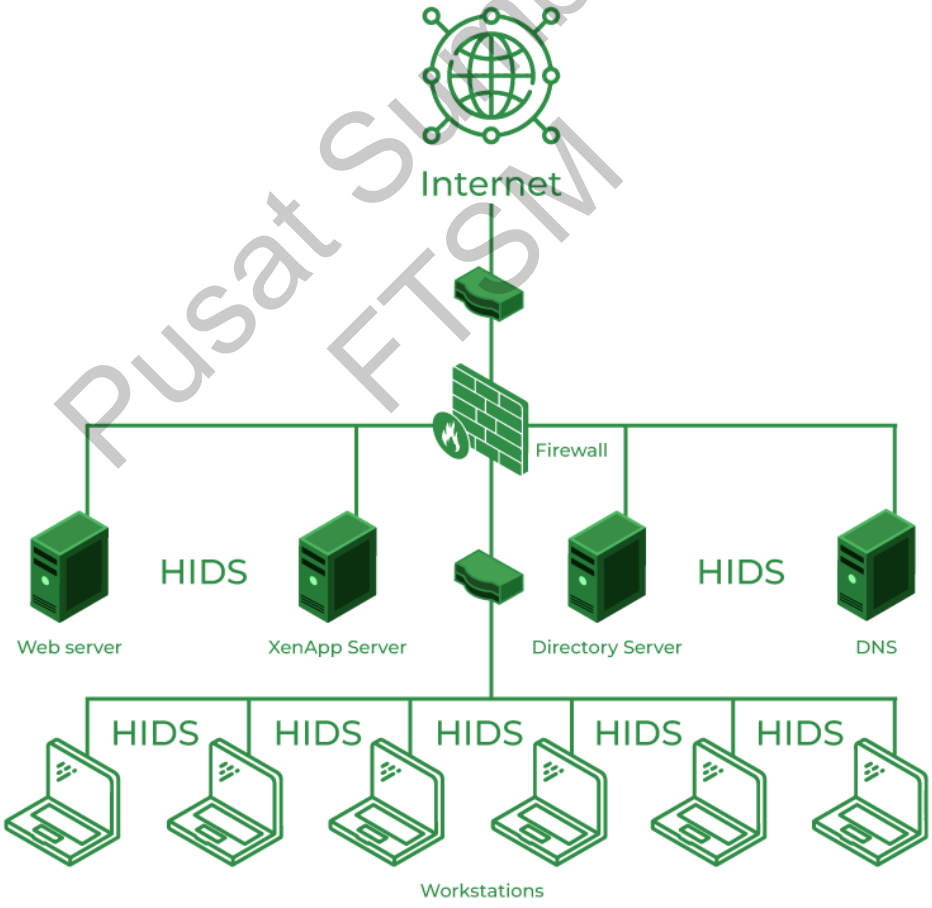


Figure 1.3 Host-based Intrusion Detection System (GeeksForGeeks, 2019)

1.2.2 Overview of Intrusion Detection Technology

IDS systems detect threats by continuously monitoring packets traversing the network. They're adept at identifying both malicious and abnormal activities originating from internal and external sources. In the figure 1.4 below summarize the overview of the IDS for types, techniques, and detection mechanisms (Aljanabi et al., 2021).

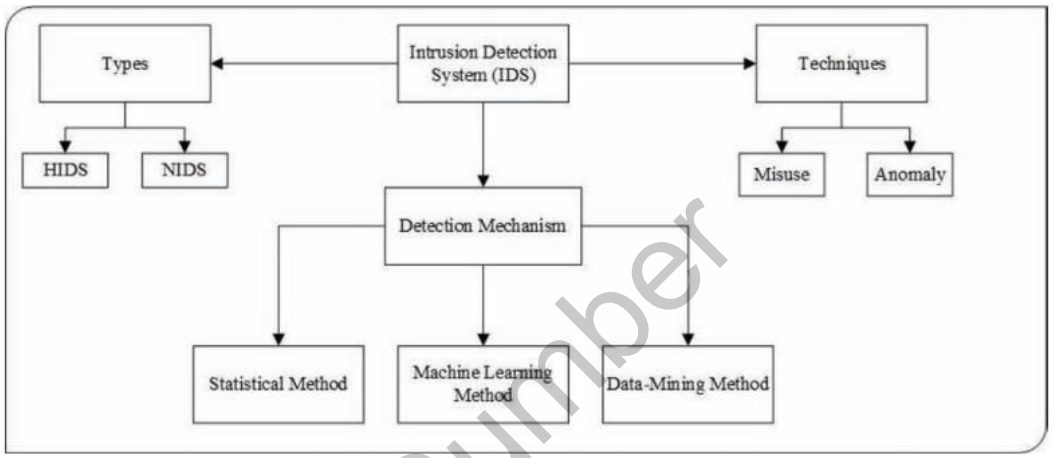


Figure 1.4 Overview Intrusion Detection System (IDS) (Aljanabi et al., 2021)

However, IDS encounters challenge due to the inherently uneven distribution of data and the immense volumes of network traffic, posing obstacles to effective threat detection and analysis.

IDS actively monitors information sources like networks and computers, focusing on reporting any unauthorized activities or accesses. It collates and scrutinizes data from multiple network sources and systems, conducting thorough analyses to identify potential threats and attacks. Figure 1.5 provides a condensed overview depicting how IDS operates within different deployment environments and employs various detection techniques. The field of intrusion detection encompasses a wide array of methods, including machine learning-based approaches, data mining methodologies, and statistical techniques. Ranging from tiered monitoring configurations to integration with antivirus software, IDS exists in diverse implementations, facilitating comprehensive surveillance of network traffic across various infrastructures (Ahmed et al., 2022).

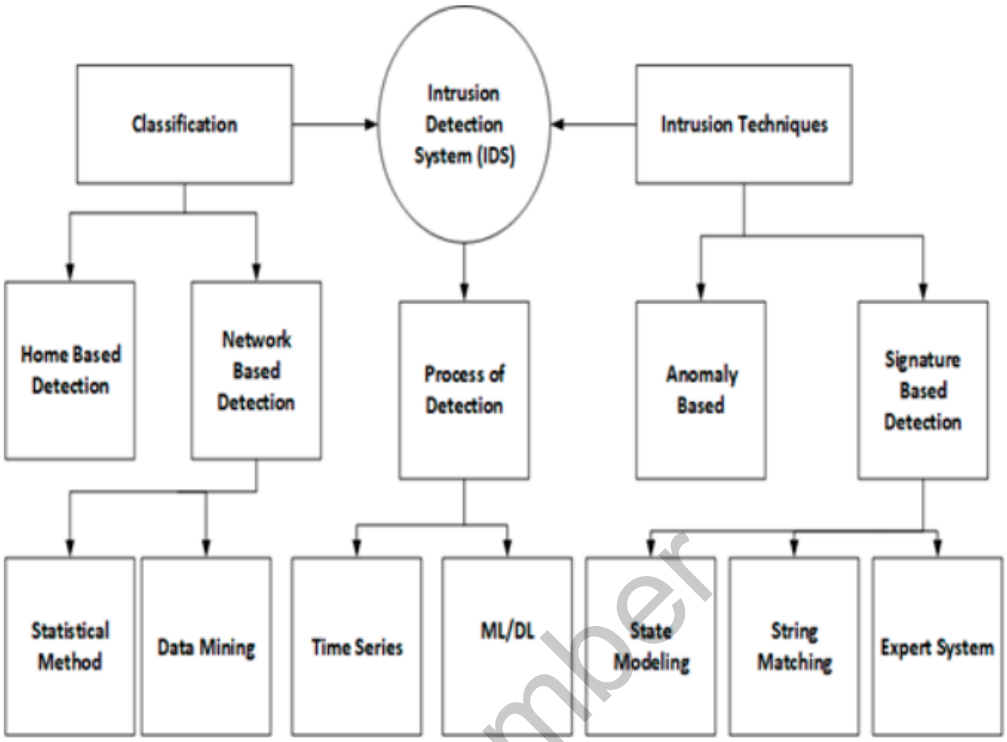


Figure 1.5 IDS Deployment Environments and Implemented Techniques of Detection (Ahmed et al., 2022)

1.2.3 Intrusion Detection Classification

Unauthorized access attempts aimed at infiltrating a computer network constitute intrusions, seeking to breach security protocols. IDS, short for Intrusion Detection System, operates as software designed to identify and signal security breaches within a network. Its core purpose revolves around scrutinizing the behaviour of hosts and network activities to uncover and flag potentially malicious actions or activities (G.Janani Pandeeswari & S. Jeyanthi, 2022). The classification of intrusion detection systems can be delineated into distinct classes:

1. The examination of incoming network traffic falls under the purview of a system known as NIDS (Network Intrusion Detection System).
2. The vigilant monitoring of critical files within the operating system characterizes another type of system labelled as "Host-based Intrusion Detection Systems (HIDS)."
3. These classifications are further refined, with commonly used variants hinging on the fundamental principles of signature and anomaly detection.

1.2.4 Challenges With Intrusion Detection System

In today's rapidly evolving data landscape, the effectiveness of current intrusion detection methods within network security is diminishing. Adapting to the modern network terrain necessitates intrusion detection technology to evolve and harmonize with the dynamic environment. The complexity of contemporary networks poses a daunting hurdle in isolating intrusion detection features from the vast sea of data, thereby persistently plaguing intrusion detection techniques with challenges. These challenges are false alarm rate, low detection rate, unbalanced datasets, and response time.

a. Slow detection speed

In today's landscape of escalating big data, establishing a comprehensive database of characteristic behaviours faces formidable challenges. The sheer volume presents hurdles in database maintenance and updates, complicating these essential tasks. The emergence of new intrusion types, adept at clever disguises and exhibiting variability, notably diminishes detection performance, leading to a surge in missed detections and false alarms (Arshad et al. 2020).

b. Low detection rate, high false alarm rate and leakage rate

Signature or rule-based IDS typically contend with a specific level of false positive (FP) alarms, struggling to identify newly emerging attack patterns. There's an anticipation for IDSs to demonstrate precision in FP detection, but these systems often fall short in detecting innovative attack forms. ID systems relying on stateful protocol analysis showcase varied detection efficiencies, contingent upon the depth of their profile definition. However, a significant hurdle lies in maintaining current profiles, especially as protocols continuously evolve, posing a constant challenge to keep them up to date. (Aljanabi et al., 2021).

c. Unbalanced datasets

Creating anomaly IDS systems with sophisticated features poses substantial challenges, particularly in the realm of misuse detection. This complexity contributes to an elevated occurrence of false alarms, often paired with a diminished detection rate. Additionally, the prevalence of unbalanced datasets presents a significant hurdle, profoundly influencing the evaluation of the models used within these systems. (Arshad et al. 2020).

1.2.5 Deep Learning Architecture (DL)

Deep learning represents a specialized branch within the broader field of machine learning, characterized by the utilization of neural networks featuring multiple layers, commonly referred to as deep neural networks. The primary objective is to tackle intricate tasks by acquiring hierarchical representations of data. Drawing inspiration from the intricate structure and functioning of the human brain, these networks are composed of interconnected nodes arranged into layers. At each layer, information undergoes processing, and distinctive features are extracted, with the deeper layers delving into progressively abstract and intricate representations (Marcus, 2018).

Deep learning's origins date back to 1943, when Walter Pitts and Warren McCulloch devised a computer model inspired by the neural networks within the human brain. Their approach, termed "threshold logic," blended algorithms and mathematical principles to replicate cognitive processes. Subsequently, the evolution of Deep Learning has been relatively continuous, save for two notable interruptions linked to the notorious periods of Artificial Intelligence winters (Foote, 2017).

Figure 1.6 illustrates a comparison between a simple neural network and a deep learning neural network. The additional layers in deep neural networks significantly enhance computational capabilities, enabling them to achieve remarkable performance across various tasks (Gunarathna et al., 2020). Conversely, Figure 1.7 showcases the prevalent deep learning architectures used in both supervised and unsupervised learning settings (Samaya Madhavan & Tim Jones, 2021). This project will primarily emphasize supervised learning and focus on Recurrent Neural Networks (RNNs). These architectures within the scope of supervised learning offer distinct advantages for sequential data analysis, aligning with the objectives of this study.

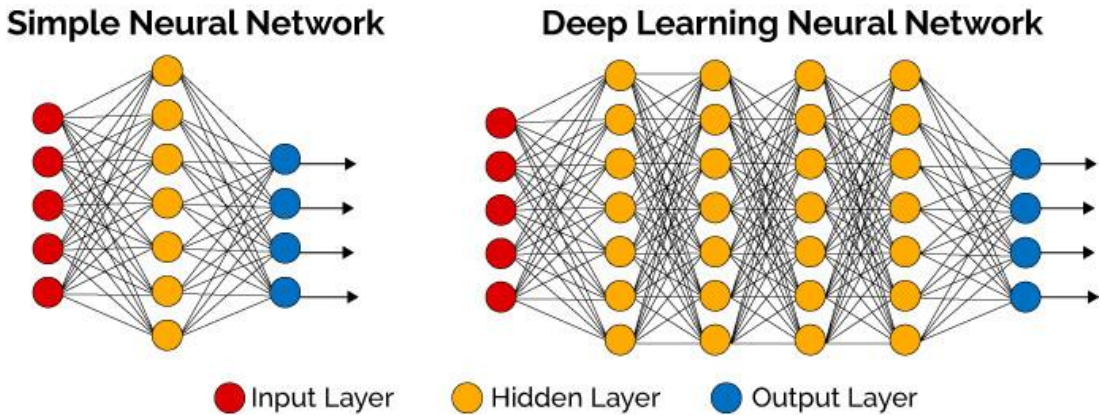


Figure 1.6 Comparison of Simple Neural Network and Deep Neural Networks (Gunarathna et al., 2020)

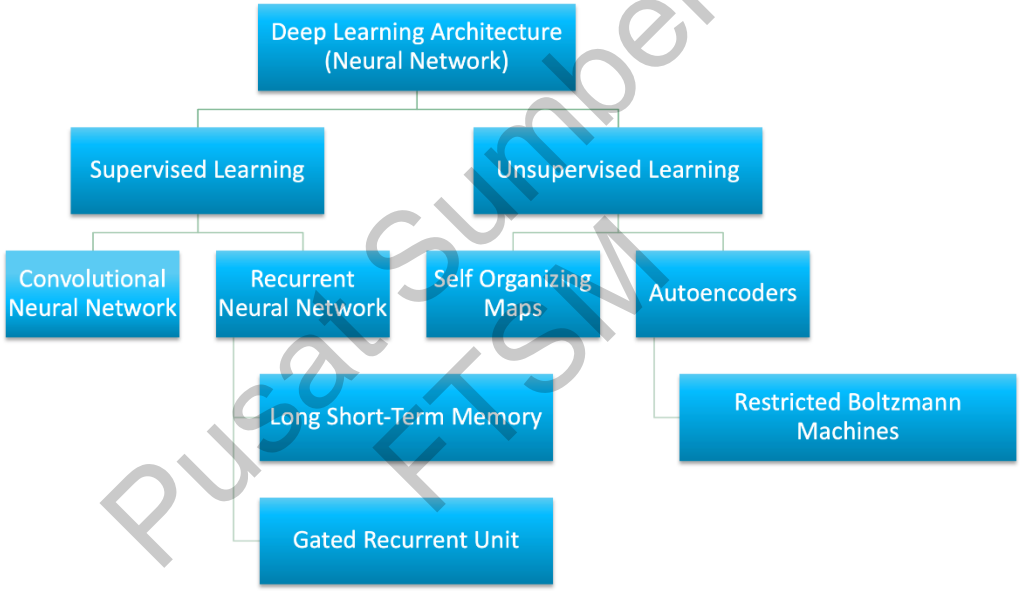


Figure 1.7 Deep Learning Architecture (Samaya Madhavan & Tim Jones, 2021)

1.2.6 Deep Learning and Intrusion Detection System (DL + IDS)

(CANER et al., 2022) enhance the deep learning technologies to the intrusion detection systems. It is an automated network monitoring tool that scans network data for known attack patterns or abnormalities to identify malware. Network traffic is observed by network intrusion detection systems (IDS) to identify traffic patterns that might endanger systems or networks. The investigation of deep learning algorithms is prompted by the potential for transmission speed reduction caused by filtering traffic that conforms with rules. The fact that it might be challenging to detect this type of IDS in a system implies that an attacker is frequently unaware that NIDS are there.

Deep learning techniques are considered since filtering out traffic that conforms with rules may slow down communication. The fact that this type of IDS may be challenging to detect in a system implies that an attacker may not even be aware that NIDS is observing his activities. The fact that this type of IDS analyses such high amounts of traffic, however, has a few drawbacks, including the possibility of inaccuracy and the production of too many false positives or even some false negatives.

The purpose of this project is to research and compare deep learning RNN models and find all the publications and methods that previous researchers have employed. It will define further information regarding the DL model and dataset. The sheer volume of ongoing network attacks makes it essential to look at this technology right now. There are two kind of network intrusions: passive and aggressive. When malicious actors get unauthorized access to a network, they may monitor it, keep an eye on it, and steal sensitive information without making any modifications. Active network attacks involve changing, encrypting, or erasing data. Network attacks that have an effect include DDoS, Man-in-the-Middle attacks, and SQL Injection.

Nevertheless, the development of AI-based solutions demands careful consideration, as several crucial factors come into play when constructing a generalized Intrusion Detection System (IDS) model. Factors such as the volume of data employed for training, performance metrics derived from unseen testing data to ensure avoidance of overfitting, the duration of the training process, the complexity of the model, various learning approaches, attributes inherent in the dataset, the computing environment, and the resources required for deployment all contribute to the nuanced landscape of

constructing an effective IDS model. Each of these elements necessitates thoughtful attention to ensure the robustness and reliability of the AI-based solution. Figure 1.8 below shows the IDS perspective on deployment or detection methods (Ahmad et al., 2020).

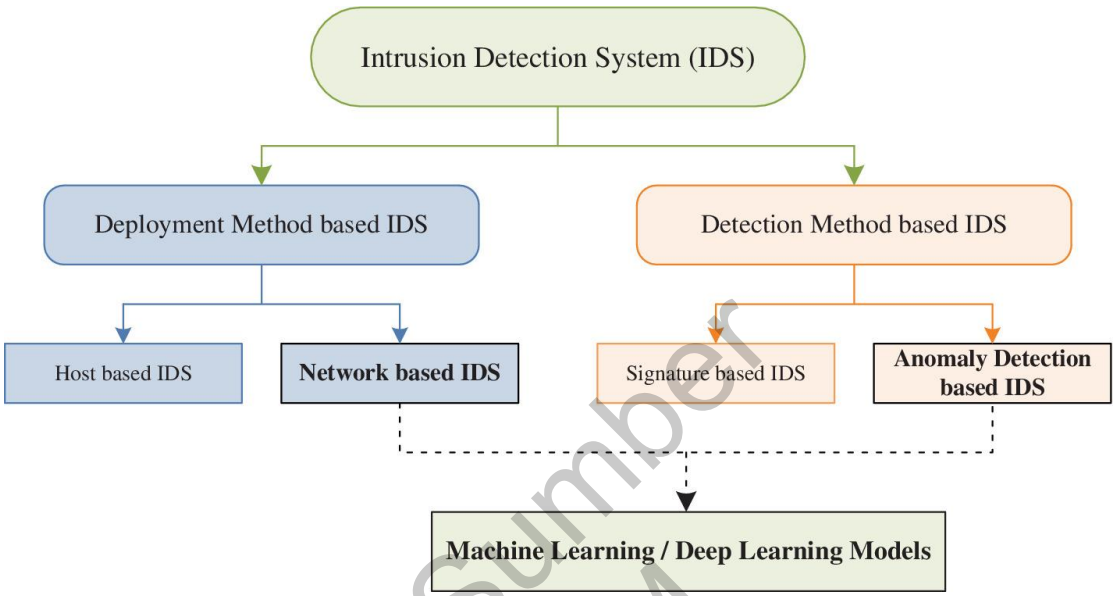


Figure 1.8 Intrusion Detection System Classification Taxonomy (Ahmad et al., 2020)

1.3 PROBLEM STATEMENT

Network intrusion, defined as the surreptitious and unauthorized access or manipulation of computer networks, represents a pervasive and grave predicament in today's digital milieu. From financial institutions protecting their transaction records to healthcare providers safeguarding the sanctity of sensitive patient data, the repercussions of a network breach resonate deeply.

The Intrusion Detection System (IDS) is an influential method developed primarily to identify irregular activities within computer systems or specific applications. However, conventional IDS methods, like machine learning and pattern recognition, lack inherent AI characteristics. Consequently, their efficacy in recognizing intricate cyber-attacks within web applications, especially those that are dynamic and complex, remains limited compared to deep learning approaches. Deep learning methodologies, leveraging multilayered processing, offer enhanced accuracy, enabling more adept detection of evolving cyber threats within these systems. (Althubiti et al., 2018).

The traditional methods relied upon by the IDS included encryption-decryption processes, protocol controls, firewalls, and anti-virus software models. Although effective to a certain extent in identifying specific attacks, these methods struggled with detecting numerous other attacks, leading to a higher frequency of false positives. Notably, Denial-of-Service (DoS) attacks posed a considerable challenge, overwhelming traditional defence mechanisms. As a result, current research has shifted its focus towards integrating machine learning (ML) techniques into intrusion detection systems.

In contrast to traditional methods, machine learning (ML) techniques showcase improved identification rates while also minimizing the operational burden linked to handling extensive attacks. Among these techniques, Support Vector Machines (SVMs) excel in identifying targeted attacks within test datasets by leveraging insights gleaned from the training data. Additionally, SVMs demonstrate efficiency in memory utilization, enhancing their suitability for intrusion detection scenarios (Hnamte et al., 2023).

Through analysing this technology, it also consists of benefits and challenges of enhancement for deep learning models in IDS. Various types of network attacks have been simulated and presented in numerous datasets in the current research. New advancements for naturally distinguishing unpredictable framework utilizations are being explored. Besides, the proposed development of an interruption recognition model as an establishment for a universally useful IDS. From that point forward, experts have made and carried out different strategies for computerizing the organization IDS strategy. They have additionally reliably looked for more exact, speedier, and versatile advancements for this objective.

With the presence of the "IoT" and Large Information times, the quantity of connected gadgets is anticipated to move toward 26 billion by 2020. Because of this turn of events, the assortment and measure of network safety challenges are projected to develop. The issues in IDS are summed up in the figure 1.9. These hardships incorporate a high misleading problem rate, an unfortunate recognition rate, imbalanced datasets, and a sluggish response time (Aljanabi et al., 2021).

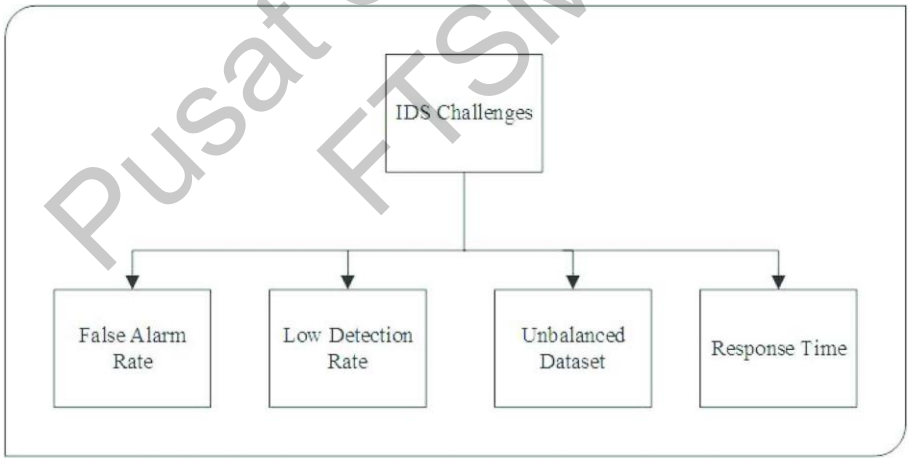


Figure 1.9 IDS Challenges (Aljanabi et al., 2021)

In recent times, researchers have advocated for expanding IDS categories by proposing a segmentation into five distinct sub-categories, which can fall under any of the previously mentioned groupings. These subclasses encompass pattern-based, rule-based, statistics-based, state-based, and heuristic-based intrusion detection systems (IDS).

To confront these challenges, researchers have turned to deep learning architectures for intrusion detection. Liu et al. (2020) introduced a deep neural network (DNN) comprising 200 hidden layers tailored for integration into an intrusion detection system. They trained the model utilizing the NSL-KDD dataset, achieving commendable detection outcomes on the training set with an accuracy of 93%. However, as research advanced, experimentation with recurrent neural network (RNN) architectures revealed a significant enhancement in accuracy metrics when compared to DNN models. This notable improvement in performance underscores the necessity for a thorough exploration and assessment of RNN models for intrusion detection tasks, especially in contrast to established DNN approaches. Consequently, there exists a critical imperative to examine and substantiate the heightened accuracy of RNN models over their DNN counterparts, thereby shedding light on the potential of RNN architectures in propelling intrusion detection systems forward. The objectives and research questions guiding our study were detailed in sections 1.4 and 1.5, framing our investigative scope and goals.

1.4 RESEARCH OBJECTIVES

i. To evaluate the performance of different recurrent neural network (RNNs) architectures, including Simple, LSTM and GRU, for intrusion detection on the NSL-KDD dataset.
ii. To investigate the impact by modifying the hypermeters, such as batch size, epochs, sequence length and parameter, on the effectiveness of RNN models in detecting network intrusions.
iii. To compare the performance of RNN models with other deep learning models commonly used in intrusion detection, such as DNN (Liu et al. 2020).
iv. To explore the effectiveness of RNN models for binary and multi-classification of intrusion attacks on the NSL-KDD dataset.

1.5 RESEARCH QUESTIONS

i. How does the performance of Simple, LSTM and GRU models compare in detecting intrusion on the NSL-KDD dataset?
ii. What is the influence of hyperparameters including batch size, epochs, sequence length and parameter, on the detection accuracy of RNN models for intrusion detection?
iii. How does the performance of RNN models compare with other deep learning models, such as DNN, in terms of detection accuracy and computational efficiency?
iv. Are there specific architectural modifications or training strategies that can enhance the performance of RNN models in accurately classifying multiple types of intrusion attacks simultaneously?

Based on the research objectives and questions, this project aims to provide insights into the efficacy of recurrent neural network models for intrusion detection and contribute to the advancement of network security measures.

1.6 SIGNIFICANCE OF RESEARCH

An intrusion detection system (IDS) serves as a vigilant tool scrutinizing network data to spot malicious activities, whether through recognized attack patterns or anomalies. This research delves into the performance evaluation of multiple deep learning-based intrusion detection and classification systems. It organizes these systems based on their employed deep learning methodologies, showcasing their efficiency in detecting intrusions. Deep learning networks showcased in this study exhibit a remarkable capability to handle diverse data types such as network traffic, system logs, and other relevant sources. This versatility enables them to identify multi-layered intrusions across network, host, and application layers, elevating the precision of intrusion detection. A comprehensive analysis of the strengths and weaknesses of each DL model was conducted, aiming to refine and optimize existing cybersecurity frameworks. The goal lies in mitigating potential cybersecurity threats within organizations.

1.7 RESEARCH SCOPE

The scope of this research project is delineated to ensure a focused and achievable exploration of RNN models (Simple-RNN, LSTM and GRU) in the context of network intrusion detection. The project encompasses the following key aspects:

Experiments of Recurrent Neural Network Models for Intrusion Detection on NSL-KDD Dataset:

a. Deep Learning Techniques
b. Data Sources
c. Comparative Analysis
d. Performance Metrics
e. Experimentation and Evaluation
f. Limitations
g. Future Research Considerations

1.8 THESIS ORGANIZATION

This project has been structured to 5 chapters. There are Chapter 1 reports on the project introduction, Chapter 2 reports on the literature review conducted, Chapter 3 reports on the research methodology used, Chapter 4 reports on experiments and results, followed by conclusion and future works in Chapter 5.

Chapter 1 lays the groundwork for this project, offering an encompassing introduction covering critical elements such as the research context, problem statement, objectives, and questions, along with highlighting the research's significance and delineating its scope.

Chapter 2 is the literature review section. It begins by introducing the relevant knowledge of intrusion detection, including an overview, classification, and challenges faced in intrusion detection. Then, it discusses the concepts of deep learning, common network models, and popular network frameworks. Next, it introduces commonly used datasets in IDS. Finally, we provided the comparative analysis and conclusion of chapter 2.

Chapter 3 provides a detailed description of the proposed deep learning RNN models in IDS framework. We provided the features of NSL-KDD dataset, which kinds of attack was labelled, and explained for whole of the methods will be used in our chapter 4 experiments.

Chapter 4 presents the experimental results and analysis for each of the RNN models, parameter, and different classification attacks to detect the network intrusion. The experimental environment was introduced and using the network intrusion dataset NSL-KDD for this experiment.

Chapter 5 concludes the entire results of project, highlighting the future upcoming of the study and provided an overview of future work.

CHAPTER II

LITERATURE REVIEW

2.1 INTRODUCTION

The realm of network security has long emphasized the significance of detecting intrusions to identify unauthorized access within secure internal networks. Network Intrusion Detection Systems (NIDS) are assembled by tapping into network equipment, utilizing devices like routers, switches, and Terminal Access Points (TAPs). These instruments serve as vigilant overseers, monitoring network breaches and violations of established policies. Many enterprises employ NIDS in tandem with firewalls and application firewalls to fortify web servers sharing the same network and system. Recent advancements in cyber-attacks have outsmarted conventional security measures by leveraging unconventional tactics such as encoding and obfuscation. To combat these evolving threats, we've turned to AI-powered IDS systems, capable of spotting and flagging variant attacks that easily evade the detection of traditional signature-based NIDS. (Hnamte et al., 2023).

Current research heavily focuses on integrating machine learning (ML) methods into intrusion detection systems. These ML techniques show promise in boosting identification rates while reducing the management overhead associated with handling extensive attacks. Support Vector Machines (SVMs) particularly excel in recognizing specific attacks within test datasets by leveraging training data properties, all the while maintaining efficient memory usage. In the realm of IDS models, SVMs stand out for their use of hyperplanes and kernel functions, enabling them to effectively categorize different attack types within datasets. Machine learning classification algorithms, including logistic regression, decision tree, random forest, K-nearest neighbor (KNN), and support vector machine (SVM), prove to be apt choices for intrusion detection.

Among these, the K-nearest neighbor (KNN) design stands out as a straightforward, rapid, and efficient solution in addressing the complexities of intrusion detection.

However, by using deep learning architectures, it able to increase the detection efficiency, reduces false positives, and eliminates the need for manual feature engineering. It enables intelligent identification of attack features, automating the detection of potential security threats (Zhang et al., 2018).

2.2 DEEP LEARNING MODELS

Deep learning (DL) is characterized by its ability to learn hierarchical representations of data through architectures featuring multiple hidden layers. The proliferation of high-performance computing facilities has propelled the popularity of deep learning techniques, particularly those employing deep neural networks. This section provides an overview of prevalent deep learning models employed in Intrusion Detection Systems (IDS), including the Deep Neural Network (DNN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), as well as variations like RNN-LSTM and RNN-GRU. These diverse models offer a spectrum of capabilities in capturing intricate patterns and relationships within data, contributing to the effectiveness of deep learning in the context of intrusion detection. Figure 2.1 shows the relationship between AI, ML and DL (Shiri et al., 2023).

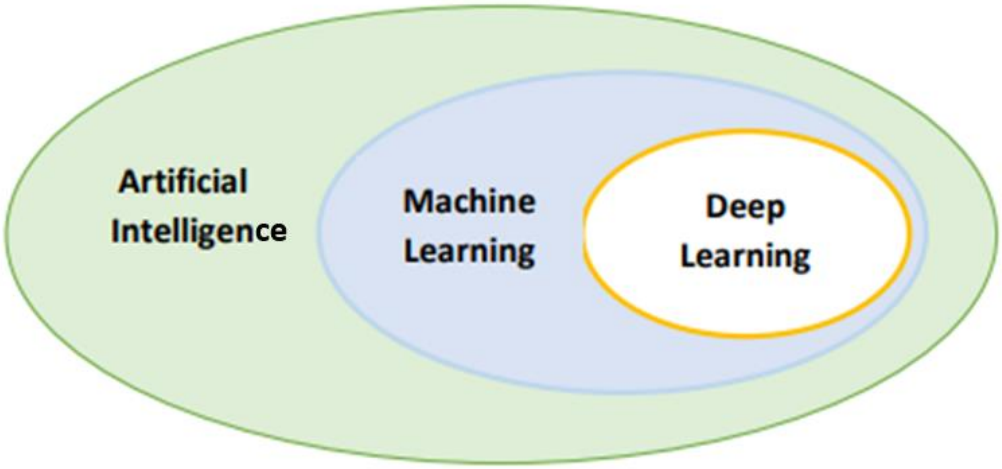


Figure 2.1 Relationship between Artificial Intelligence, Machine Learning and Deep Learning (Shiri et al., 2023)

2.2.1 Deep Neural Network (DNN)

Traditional machine learning algorithms follow a straightforward path, while deep neural networks work in layered complexity, each level learning more abstract features. Picture it like passing information from the input layer to the hidden layers, where complex computations happen. Deciding how many of these hidden layers and neurons they contain is a big challenge. Each neuron uses an activation function to process its output. In "deep" learning, there's a focus on multiple hidden layers. The output layer gives the final data, and training continues through epochs until accuracy reaches an acceptable level. (Vigneswaran et al., 2018).

a. The Model Topology

The input layer serves as the point of entry for the neural network's data initialization, utilizing 125 nodes that correspond to the pre-processed dataset's features.

Situated between the input and output layers, the hidden layers undertake computational tasks within the model. Employing two hidden layers, the system consists of 50 neural nodes in the first layer and 30 neural nodes in the second layer, chosen based on training outcomes to optimize performance.

Lastly, the output layer generates the results, distinguishing between normal activity and various attack types detected by the model. The connections of the nodes stated in figure 2.2 (Maithem & Al-sultany, 2021).

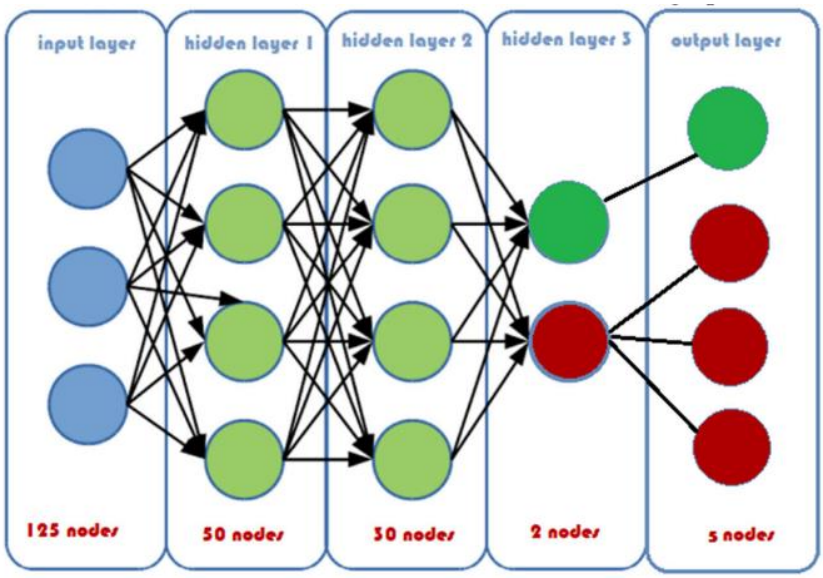


Figure 2.2 Proposed Deep Neural Network Topology (Maithem & Al-sultany, 2021)

In the figure 2.3 below is shows the deep neural network architecture for anomaly-based IDS (Tama & None Kyung-Hyune Rhee, 2017) and figure 2.4 illustrate deep neural network architecture for multi-classification (Tang et al., 2020).

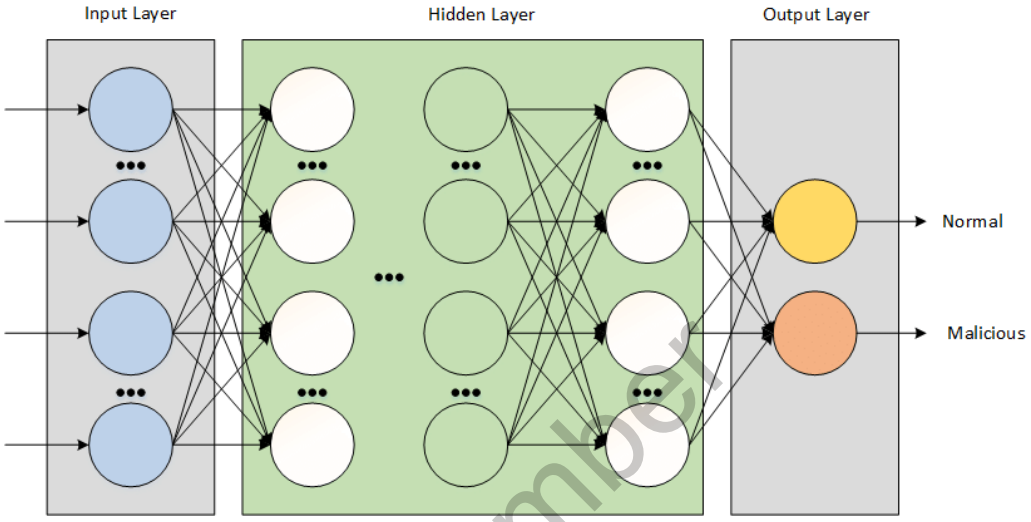


Figure 2.3 Deep Neural Network Architecture for Anomaly-based IDS (Tama & None Kyung-Hyune Rhee, 2017)

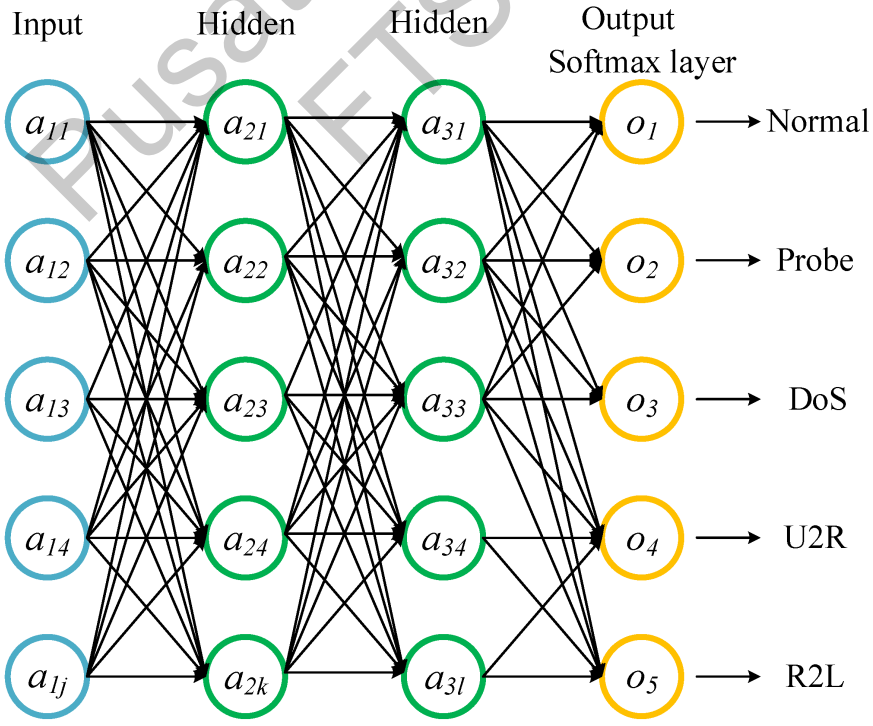


Figure 2.4 Deep Neural Network Architecture for Multi-Classification (Tang et al., 2020)

2.2.2 Convolutional Neural Network (CNN)

Advancements in computational resources have propelled the long-established convolutional neural networks (CNN) into a realm of significant progress within deep learning. CNN's unique architecture significantly enhances data representations, notably in image recognition and sentence modelling, yet its application remains notably absent in intrusion detection because of complexity of network traffic patterns and imbalanced data. A typical CNN structure for visual analysis involves an input and output layer, along with multiple hidden layers, including convolutional, pooling, fully connected, and normalization layers. Within CNN, convolutional layers execute convolution operations on preceding inputs, forwarding the outcomes to subsequent layers in the network. In CNNs, a convolutional neuron handles data within its receptive field through a convolution operation, focusing solely on that area. To optimize memory and enhance performance, weights are shared across receptive fields within the same filter. Pooling layers, a key component, come in two primary types: max pooling, which selects the highest value, and average pooling, which computes the mean value within defined regions (Ding & Zhai, 2018). In figure 2.5 below is a CNN model IDS (Liao et al., 2024).

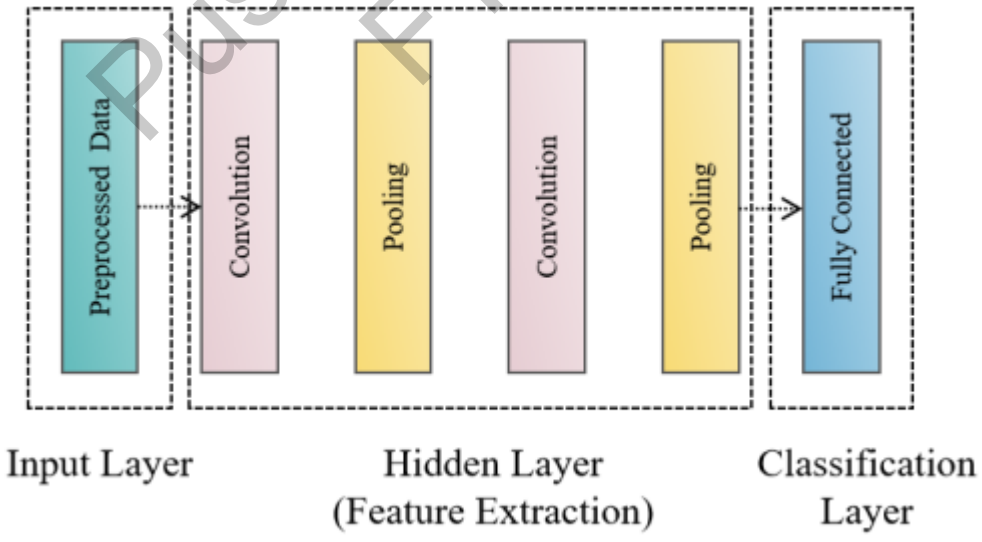


Figure 2.5 CNN model Intrusion Detection System (Liao et al., 2024)

2.2.3 Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNNs) stand out among deep learning models for their inherent memory, allowing them to grasp sequential patterns effectively. Unlike conventional neural networks, which view inputs in isolation, RNNs factor in the temporal sequence of inputs, proving beneficial for tasks reliant on sequential data. Using a looping mechanism, RNNs iteratively process elements within a series, leveraging both the present input and previous computations to generate the current output (Farhad et.al, 2023)

In contrast to simpler neural networks like Multilayer Perceptron's (MLPs), Recurrent Neural Networks (RNNs) aren't constrained by one-way information processing. They possess the unique ability to cycle through various layers, allowing for the retention of temporary information for subsequent use. Figure 2.6 illustrates the composition of a standard RNN (sRNN) or Simple RNN, where *NN* signifies a standard neural network, *xp* denotes the input, and *hp* represents the output. RNNs are categorized as deep neural networks owing to their utilization of multiple layers in information processing. Furthermore, Figure 2.7 presents an unrolled depiction of a standard sRNN, highlighting the inherent multi-layered architecture within RNNs (Kasongo, 2022).

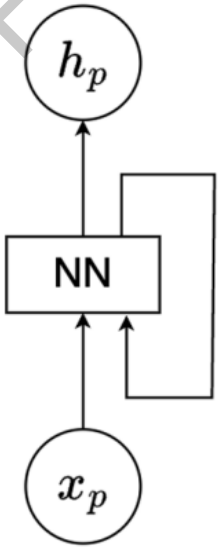


Figure 2.6 Standard RNN (Kasongo, 2022)

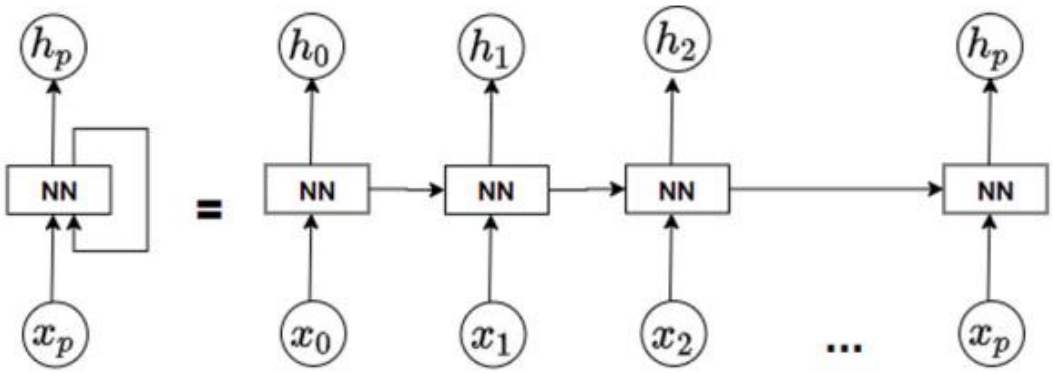


Figure 2.7 A unrolled standard RNN (Kasongo, 2022)

2.2.4 Long Short-term Memory Network (LSTM)

In 1997, Hochreiter and Schmidhuber introduced the Long Short-Term Memory (LSTM) network to solve the vanishing gradients problem found in standard RNNs. Unlike regular RNNs, LSTM replaces the unit cell with a memory cell, depicted in Figure 2.8. Since its inception, various versions of LSTM have emerged, but the architecture described here is the most used. At its core, an LSTM unit contains two main elements: an internal state and gates. This upgraded RNN, LSTM, effectively handles the challenge of retaining long-term patterns in sequences of data (Zargar, 2021).

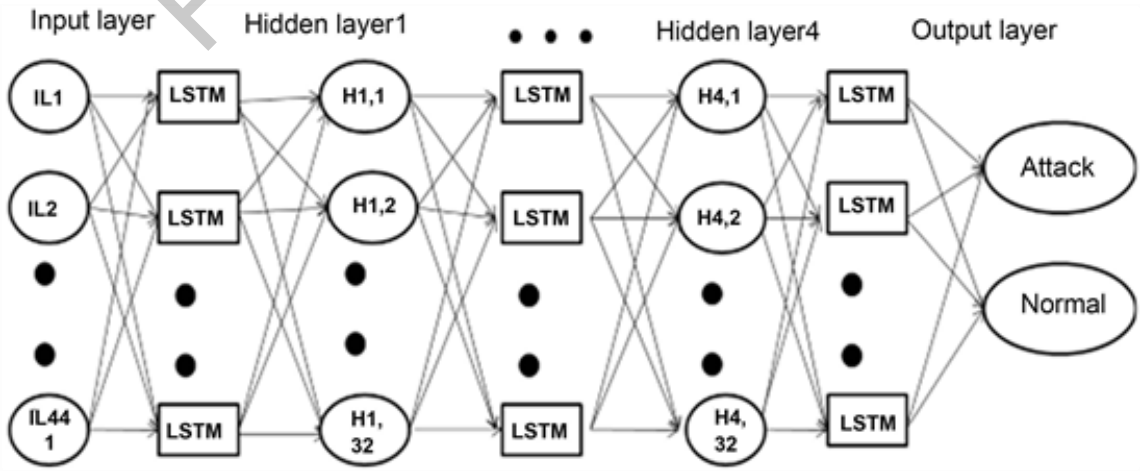


Figure 2.8 Long Short-Term Memory (LSTM) (Zargar, 2021)

2.2.5 Gated Recurrent Unit (GRU)

The GRU, introduced in 2014 by Cho et al. and Chung et al., represents a simpler alternative to the LSTM. Its internal structure is less complex than the LSTM, making it easier to train due to reduced computational demands. This simplicity is achieved through two primary modifications: first, combining the input and forget gates into a singular update gate, and second, merging the internal/cell state with the hidden state. Within the architecture of a GRU unit, three fundamental elements play a pivotal role: the update gate, the reset gate, and the existing memory content. These gates endow the GRU with the capability to selectively update and harness information from preceding time steps. This unique ability equips the GRU to effectively capture and maintain long-term dependencies within sequences, contributing to its proficiency in handling intricate data relationships over time (Farhad et.al, 2023). Figure 2.9 present the gated recurrent unit implementation (Kilinc & Yurtsever, 2022).

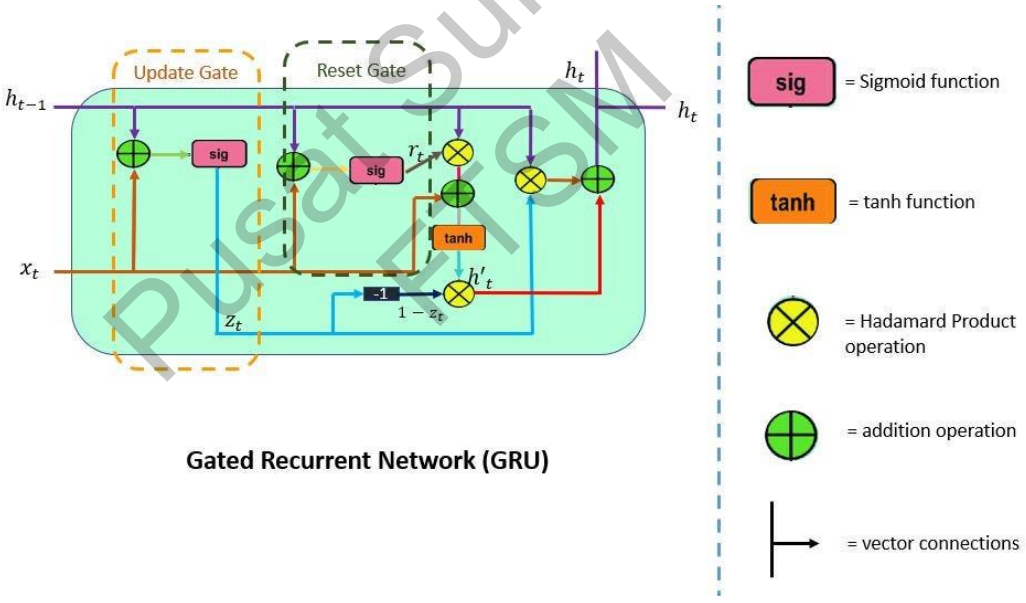


Figure 2.9 Gated Recurrent Unit (GRU) Implementation (Kilinc & Yurtsever, 2022)

Figure 2.10 below shows the RNN, LSTM and GRU models for intrusion detection system (Toharudin et al., 2021), and table 2.1 is the short comparison for RNN models.

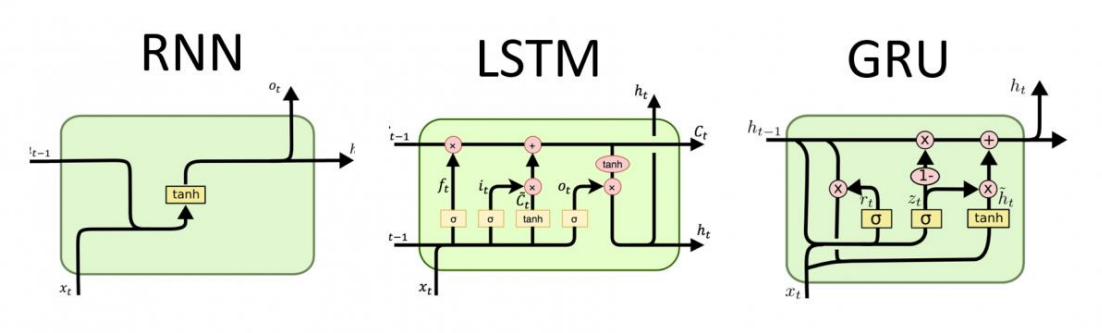


Figure 2.10 RNN, LSTM and GRU models for Intrusion Detection System (Toharudin et al., 2021)

Table 2.1 Comparison of RNN models

Parameters	RNNs	LSTMs	GRUs
Structure	Simple	More complex	Simpler than LSTM
Training	Can be difficult	Can be more difficult	Easier than LSTM
Performance	Good for simple tasks	Good for complex tasks	Can be intermediate between simple and complex tasks
Hidden state	Single	Multiple (memory cell)	Single
Gates	None	Input, output, forget	Update, reset
Ability to retain long-term dependencies	Limited	Strong	Intermediate between RNNs and LSTMs

In summary, deep learning models for DNN, CNN and RNN are the most used for the IDS experiments. We would like to choose RNN models (Simple RNN, LSTM, and GRU) for further investigation and comparison in intrusion detection systems (IDS) due to the inherent capability to effectively analyze sequential data, such as network traffic. Besides that, LSTM and GRU variants address the limitations of traditional RNNs by enabling the capture of long-term dependencies in the data while selectively retaining relevant information. The models flexibility in handling variable-length sequences and efficient feature learning make them promising candidates for identifying subtle intrusion patterns and anomalies in network traffic data, thus enhancing the effectiveness of cybersecurity measures.

2.3 NETWORK INTRUSION DETECTION DATASET

2.3.1 NSL-KDD

The NSL-KDD dataset, abbreviated from "Network-based System for Learning," was introduced as an enhanced version to bolster intrusion detection research within cybersecurity. This dataset serves as a prevalent resource in the field, addressing limitations present in the original KDD Cup 1999 dataset, a common benchmark for intrusion detection system studies. Created after eliminating redundant and duplicate records from the KDD Cup, it comprises a refined selection of essential records for experimentation. The NSL-KDD dataset comprises a total of 37 distinct attacks, distributed with 27 attacks in the testing dataset and 23 in the training dataset. It retains the same number of features as the original KDD Cup, boasting 41 features distributed among 5 attack classes. The array of attacks within the dataset includes a standard class representing normal behaviour alongside four distinct attack types categorized as Probe, Denial of Service (DoS), User to Root (U2R), and Remote to Local (R2L). This diverse categorization enables a broad spectrum of intrusion detection experiments across varied attack scenarios. Table 2.2 and 2.3 below present the comparison of the KDDCup99 and NSL-KDD dataset (Bala, 2019).

Table 2.2 Comparison of KDDCup99 and NSL-KDD Dataset

Attack Category	**Full Dataset**	**10% Dataset**			
	KDDCup 99	**KDDCup 99**		**NSL-KDD**	
	Train	**Train**	**Test**	**Train**	**Test**
Normal	972,780	97,278	60,593	67,343	9,711
DoS	3,883,370	391,458	229,853	45,927	7,460
Probe	41,102	4,107	4,166	11,656	2,421
R2L	1,126	1,126	16,189	995	2,885
U2R	52	52	228	52	67
Total	4,898,430	494,021	311,029	125,973	22,544

Table 2.3 Count of Normal and Malicious Data in NSL-KDD

Dataset	**Normal**	**Malicious**	**Total**
KDD Train^{+}	67343	58630	125973
KDD Test^{+}	9711	12833	22544

2.3.2 UNSW-NB15

The introduction of the UNSW-NB15 dataset in 2015 within the Cyber Security Lab at the Australian Centre (ACCS). It was another version of dataset compare with previous datasets like KDDCup 99 and NSL-KDD. Crafted with a specific focus on refining Network Intrusion Detection System (NIDS) models, it covers a wide range of nine distinct attack types (Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms). With an extensive array of 49 distinct features, this dataset provides a comprehensive collection of both regular and attack activities, meticulously categorized across a database of over two million records. In its original composition, it hosts 2,540,044 records, enabling the classification of normal network traffic alongside various types of network attacks (Mijalkovic & Spognardi, 2022). The dataset underwent partitioning into distinct training and testing sets, culminating in 175,341 records allocated for training purposes and 82,332 records reserved for testing. This strategic division aimed to ensure a balanced representation of diverse attack types and normal network traffic across both sets. Such a balanced distribution bolstered the model's ability to learn effectively and generalize its understanding across a spectrum of data scenarios. In table 2.4 presented the train and test connection records of partial dataset of UNSW-NB15 (Yee Mon Thant et al., 2023).

Table 2.4 Training and Testing Connection Records of Partial Dataset of UNSW-NB15

Attack	Train	Test	Total	Description
Normal	56,000	37,000	93,000	Normal connection
Fuzzers	18,184	6,062	24,246	Instances of attacks encompass activities such as spam-related intrusions, breaches involving HTML files, and incursions through port scans.
Analysis	2,000	677	2677	The analysis encompasses the investigation of specific network information by conducting scans, examining aspects such as port scanning, potential HTML file penetrations, and the identification of spam-related activities.
Backdoors	1,746	583	2329	A backdoor refers to a type of malware designed to illicitly gain unauthorized access to a computer system or network.

… to be continued

… continuation

DoS	12,264	4,089	16,353	The primary goal of an intruder is to disrupt network resources, rendering them inaccessible to authorized users.
Exploits	33,393	11,132	44,525	An attacker identifies weaknesses in an operating system or data and uses that information to exploit vulnerabilities.
Generic	40,000	18,871	58,871	Attacks on block ciphers involve hashing numerous messages to prevent the same hash from appearing again, aiming for collision resistance.
Reconnaissance	10,491	3,496	13,987	Attackers exploit system weaknesses by scanning traffic or sniffing packets to identify vulnerabilities.
Shellcode	1,133	378	1511	The term "payload" refers to a small segment of a program, often utilized in exploiting vulnerabilities within software systems.
Worms	130	44	174	Worms have the capability to self-replicate and propagate through computer networks, distributing themselves to other systems within the network infrastructure.

2.3.3 CICIDS2017

The Canadian Institute for Cyber-security recognized the limitations of older datasets, prompting them to develop newer ones like CICIDS2017. These older datasets often lack diversity in traffic, fail to encompass comprehensive features and metadata, and suffer from insufficient volumes. They might overlook the variety of cyber-attacks or anonymize traffic, disregarding crucial data payloads. CICIDS2017 come out as the most comprehensive and current dataset, addressing these shortcomings by offering a more complete representation of cyber threats. (Sharafaldin et al., 2018). The CICIDS2017 dataset satisfies all the essential criteria required to construct a precise benchmark dataset, encompassing updated attack methodologies like DoS, DDoS, Brute Force, XSS, SQL Injection, Infiltration, Portscan, and Botnet (Azzaoui et al., 2021). Over a span of five days, CICIDS2017 compiled data within a fully configured network environment, incorporating diverse devices like Modems, Firewalls, and Routers. To replicate authentic traffic, researchers meticulously analyzed human interaction behaviors, crafting genuine benign background traffic. The dataset encapsulates an array of cyber threats, including Brute Force FTP, Brute Force SSH,

Heartbleed, Botnet, DoS, DDoS, Web Attacks, and Infiltration Attacks. These attacks are defined by an extensive set of 85 numeric and nominal features, providing a detailed characterization of each type of threat (Azzaoui et al., 2021). The detailed information of CICIDS 2017 dataset is reported in table 2.5 below (Vinayakumar et al., 2019).

Table 2.5 CICIDS 207 Train and Test Dataset

Attack	Train	Test	Total	Description
Normal	60,000	20,000	80,000	Normal connection records
SSH-Patator	5,000	897	5,897	Secure shell - Representation of brute force attack
FTP-Patator	7,000	938	7,938	File transfer protocol - Representation of brute force attack
DoS	6,000	2,000	8,000	The intruder's objective is to disrupt network resources, rendering them inaccessible to authorized users.
Web	2,000	180	2,180	Related to web attacks
Bot	1,500	466	1,966	Bot owners exert control over hosts, manipulating them to execute diverse tasks like data theft, dissemination of spam, and various other activities.
DDoS	6,000	2,000	8,000	A Distributed Denial of Service (DDoS) attack is an intentional effort to disrupt services by overwhelming them with traffic from various origins. This disruptive tactic relies on harnessing a botnet, a network of compromised devices, to flood a target with an excessive volume of requests or data, rendering the services inaccessible to legitimate users.
Port Scan	6,000	2,000	8,000	A port scan helps identify open ports associated with specific services, allowing attackers to gather information about the sender and receiver's listening details.

2.3.4 CSE-CIC-IDS2018

The CSE-CIC-IDS2018 dataset, a collaboration between the Canadian government's CSE, the Canadian Institute for Cybersecurity (CIC), and Amazon Web Services (AWS), stands out as the most extensive and latest public intrusion detection dataset available. Spanning ten days, it captures genuine network attacks within its network topology, cataloging both benign and attack traffic in CSV files. With a collective size of 6.41 GB spread across 10 files, it contains a whopping 16,233,002 datasets. Interestingly, this vast dataset lacked predefined training and testing divisions due to its size and redundancies. Researchers encountered variations in data processing techniques and overall data quantities. For instance, some studies opted for 40,000 benign data samples out of the total 13,484,708, along with 20,000 attack data samples, utilizing nine out of the ten available files for their experiments. The details of attack types of CSE-CIC-IDS2018 reported in table 2.6 (Wang et al., 2023).

Table 2.6 List of Attack Types of CSE-CIC-IDS2018

Attack	Training-Validation Set	Testing Set	Attack Name
Benign	7,003,032	1,824,935	Normal
Bruteforce	75,434	18,619	FTP-Bruteforce and SSH-Bruteforce
DoS	156,525	39,443	DoS-GoldenEye, DoS-Slowloris, DoS-SlowHTTPTest, and DoS-Hulk
Web Attack	522	136	Brute Force-Web, Brute Force-XSS, and SQL Injection
Infiltration	60,382	19,379	Infiltration
Botnet	114,647	28,786	Bot
DDoS	618,384	154,529	DDoS attacks-LOIC-HTTP, DDoS-LOIC-UDP, and DDOS-HOIC

2.3.5 Commonly Used Dataset in IDS Based on Various Deep Learning

In this section, we analyzed most of the IDS used which of the datasets in recent years. We provided the details and references for some of the deep learning models selected which of the subsequent datasets in Table 2.7.

Table 2.7 Commonly used Models and Datasets in Intrusion Detection System

Model	Datasets					
	KDD Cup99	**NSL-KDD**	**UNSW-NB15**	**CICIDS2017**	**CSE-CIC-IDS2018**	**Others**
DNN	(Vigneswaran et al., 2018) (Maithem & Al-sultany, 2021)	(Hsieh & Su, 2021) (Almejarb et al., 2023) (Mijalkovic & Spognardi, 2022)	(Ashiku & Dagli, 2021) (Mijalkovic & Spognardi, 2022)			
CNN	(Vinayakumar et al., 2017)	(Vinayakumar et al., 2017) (Ding & Zhai, 2018) (Lokesh Karanam et al., 2020)			(Hagar & Gawali, 2022)	(Vanlalruata Hnamte & Hussain, 2023)
RNN (SimpleRNN, LSTM, GRU)		(Lokesh Karanam et al., 2020) (Muhuri et al., 2020) (Kasongo, 2022) (Ibrahim & Elhafiz, 2023) (Zhang et al., 2023)	(Kasongo, 2022) (Yee Mon Thant et al., 2023)		(Hagar & Gawali, 2022)	(Althubiti et al., 2018)
LSTM-AE				(Hnamte et al., 2023)	(Hnamte et al., 2023)	
ML	(Behrooz Sezari et al., 2018)	(Belgrana et al., 2021)				

In the summarizing table 2.7 above, we can see that the most popular datasets used in IDS are KDD Cup 99, NSL-KDD and UNSW-NB15. NSL-KDD still is the most widely applied and researched network intrusion detection dataset. There are some reasons why NSL-KDD is the best option for IDS. The first reason is because NSL-KDD dataset is an upgraded version of KDD Cup 99 dataset, it contains approximately 130,000 network connection records, many network traffic samples, and different types of attacks. The second reason is compared with other IDS datasets, NSL-KDD is considered not too large and enough time for training. The last reason is because NSL-KDD dataset combines with the real network traffic data, which can help to simulate intrusions in realistic network environments.

2.4 CURRENT RESEARCH IN INTRUSION DETECTION SYSTEM

a. Big data, IoT and AI for a smarter future (Ashiku & Dagli, 2021)

Due to inherent vulnerabilities, ensuring secure communication exchanges necessitates robust cybersecurity measures. Deep learning architectures are now pivotal in crafting adaptable and resilient network intrusion detection systems (IDS) capable of identifying and categorizing diverse network threats. Leveraging the UNSW-NB15 dataset, a sophisticated NIDS model was employed, simulating genuine network communication behaviours and attack scenarios. Through a deep learning classification architecture and a semi-dynamic hyperparameter tuning technique, significant enhancements were observed in comparison to other deep learning-based network IDS solutions. Notably, the proposed approach exhibited substantial improvements in multiclass models, achieving an impressive overall accuracy of 95.4% for pre-partitioned multiclass classification and 95.6% for user-defined multiclass classification. Figure 2.11 shows the detection rates for the types of each attack.

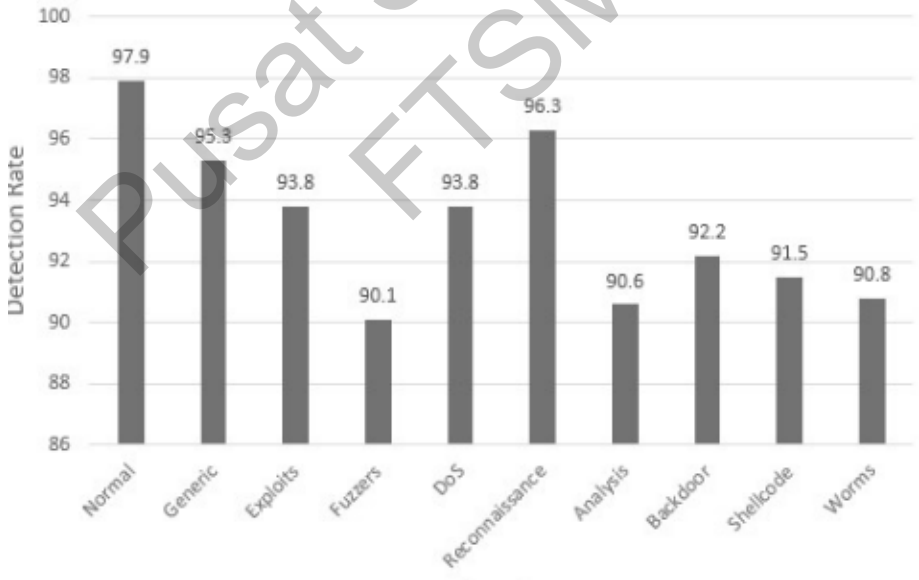


Figure 2.11 Detection rates for the types of attacks (Ashiku & Dagli, 2021)

b. Deep Learning Approach for IDS in Medical Things (Chaganti et al., 2022)

Utilizing Internet of Medical Things (IoMT) technology has proven instrumental in extending doctors' reach to more patients and enhancing real-time patient monitoring and diagnostics, potentially saving lives. Nonetheless, the vulnerability of IoMT devices to cyber-attacks poses significant concerns regarding security and privacy. Given their constrained computation and memory capabilities, implementing robust security measures directly on IoMT devices remains challenging. To address this, a novel approach employing a deep neural network (PSO-DNN) has been proposed for an efficient and precise intrusion detection system within the IoMT framework. This PSO-DNN methodology showcases a remarkable 96% accuracy in detecting network intrusions by integrating network traffic and patient sensing datasets. The study introduces a strategic feature selection technique using PSO-DNN, augmenting the performance of the intrusion detection system in IoMT and proposing a DNN-based deep learning model for improved security.

c. Network Threat Detection Using Deep Learning in SDN-Based Platforms (Ahmed et al., 2022)

SDN (software-defined networking) has ushered in a revolution in network innovation by allowing network control from a single point and providing an overview of organisation security. Network interruption detection frameworks (NIDS) detect and prevent network outages while also ensuring the organization's trustworthiness, accessibility, and privacy. Despite significant progress in NIDS, there is still need for improvement in terms of minimizing fake issues and enhancing risk location accuracy.

This research was used in SDN-based NIDS to address network security issues and examine NID frameworks based on SDN, for example, lightweight DDoS flooding attack, anomaly location, DDoS assault identification, and interruption recognition. In the exploration, four management techniques were examined. In figure 2.12 present the DL approaches for network intrusion detection.

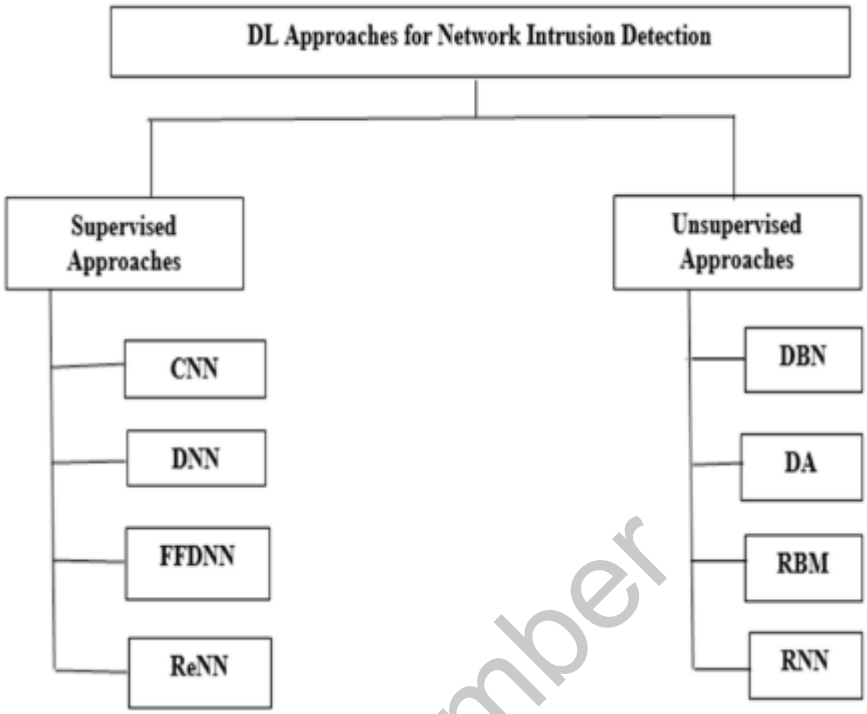


Figure 2.12 DL Approaches for Network Intrusion Detection (Ahmed et al., 2022)

SDN-based NID frameworks were proposed, and the DNN DL calculation was utilized. IDS was created to classify digital assaults utilizing DNN calculations. When contrasted with ML strategies, it accomplishes identification precision of up to 96.3%.

The FFDNN DL calculation was utilized to recommend an interruption recognition framework. The dataset was NSL-KDD, and FFDNN was used for both preparation and testing. The trial results propose that discovery exactness can reach 99.69%. One more strategy used to do LTSM is RNN, and the dataset used was KDD Cup1999. The LSTM is prepared, and the blunder of the LSTM is used as a sign to distinguish irregularities.

In this experiment, a set comprising one non-attack and four distinct attacks was utilized for evaluation purposes, achieving a robust detection rate of 98.8%. The study proposed the implementation of an LSTM-based RNN within a multichannel IDS system, leveraging the NSL-KDD dataset. The experimental assessment showcased the LSTM-RNN's performance, achieving an impressive 99.23% accuracy alongside a false alarm rate of 9.86%. Additionally, this paper introduced a network-based IDS approach utilizing convolutional autoencoders on the CTU-UNB and Contagio-CTU-UNB

datasets. A neural network (NN) model constructed using Theano showed promising results, boasting a high accuracy rate of 99.5%.

d. Towards Data-Driven Network Intrusion Detection Systems (Maabreh et al., 2022)

On one million random network intrusions from the CSE-CIC-IDS2018 large data set, this study investigates four feature selection procedures, seven standard machine learning techniques, and the deep learning algorithm. In the massive CSE-CIC-IDS2018 data collection, this covers all potential types of network intrusions, which have been divided between 30% unknown testing and 70% feature selection and model training. There is one binary label, "Benign" or "Attack," and 68 traffic characteristics in the dataset.

e. An adaptive method and a new dataset UKM-IDS20 (Al-Daweri et al., 2021)

The UKM-IDS20 emerges as a novel dataset recommended in this research. Compiled from genuine network traffic data, it incorporates an array of 46 features covering distinct attack types including ARP poisoning, DoS, scans, and exploits. Employing rough-set theory and a dynamic artificial neural network, this dataset underwent rigorous examination and comparison against the KDD99 and UNSW-NB15 datasets. The evaluation yielded an accuracy detection rate of 94.66% and a false alarm rate of 07.57% for the UKM-IDS20.

f. Reducing the False Negative Rate in Deep Learning Based on NIDS (Mijalkovic & Spognardi, 2022)

Network intrusion detection systems (NIDS) are critical components of system security because they continually monitor the network and notify users to any unusual behaviour or incident. The purpose of this basic model was to outperform, if not outperform, the models demonstrated in cutting-edge research. This research generated various models, provided a beneficial refining method, and enhanced predictability for minority classes. The collected data demonstrated that by utilising the appropriate settings, it is feasible to achieve a satisfying trade-off between FNR, accuracy, and minority class detection.

g. Performance analysis and feature selection for NIDS Deep Learning (Caner et al., 2022)

The researcher's focus on exploring the intrusion detection and classification capabilities of diverse deep learning-based systems. To achieve this goal, 24 deep neural networks, each characterized by four unique architectures, underwent training and evaluation using the CICIDS2017 dataset. Ablation research methodology was employed to assess the utility of features, subsequently aiding in the identification of the most effective model for feature selection. The attained accuracy reached 97.58% when utilizing only nine raw features, accompanied by a notable reduction in test time per sample to 40.56 seconds. This underscores the efficiency and precision achieved through the streamlined feature selection process.

h. Intrusion Detection Method Based on Deep Learning (Tian et al., 2022)

This research finalized an in-depth examination of network traffic within a big data storage environment, focusing on performance connected to Hadoop storage. The initial phase involved a comprehensive review of big data security, incorporating diverse requirements for intrusion detection. Subsequently, the study sought to introduce a decision tree analysis model, employing a feature selection method and network feature detection. This detection mechanism was implemented through a deep learning model constructed on the TensorFlow platform. The overarching objective was to not only enhance security measures but also to identify and characterize the intricate network patterns within the context of big data storage and connectivity.

i. Apache Spark and Deep Learning Models for High-Performance NIDS (Hagar & Gawali, 2022)

To identify network threats, this study used three models: Apache Spark, CNN, and LSTM. The recommended models used the random forest strategy to reduce the size of the feature reduction from 84 to 19. Apache Spark received an F1-score of one for ten classes, 100% accuracy for the remaining classes, and F1-scores of 0.99, 0.98, 0.97, and 0.98 for the other classes. When DL models and other related studies are compared, Apache Spark is the quickest model, needing just 7.56 minutes to train.

j. Deep Learning-Based Intrusion Detection Systems (Lansky et al., 2021)

In this paper, intrusion detection technologies that have helped deep neural networks deal with invasions and destructive acts are thoroughly reviewed and classified. To do this, it first categorises deep IDS systems according to the deep learning techniques they use. After that, it analyses how each scheme aims to use deep learning techniques to recognise various types of intrusions.

k. NIDS optimized with rule-based hybrid feature selection (Ayo et al., 2020)

This research presents an innovative NIDS (Network Intrusion Detection System) that leverages deep learning models while integrating a hybrid rule-based feature selection methodology. Its framework encompasses three critical stages: detection, rule assessment, and hybrid feature selection. Interestingly, the study's findings unveiled an unexpected revelation: the quantity of selected features had no bearing on the detection accuracy of the feature selection techniques. However, a notable correlation emerged between the base classifier's performance and the number of chosen features. Impressively, with training and testing times at 1.2% and 98.8%, and execution durations of 7.17s and 3.11s respectively, the proposed strategy displayed a performance comparison featuring reduced false alarm rates and elevated accuracy rates in contrast to existing techniques in the NIDS domain. These outcomes highlight the superior efficacy of the proposed approach over established methodologies within NIDS.

l. Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) to Classify Network Attacks (Muhuri et al., 2020)

A new strategy was introduced to elevate the classification of the NSL-KDD dataset through an inventive intrusion detection method. This approach merged a genetic algorithm (GA) for precise feature selection with the incorporation of a LSTM component within RNN. This fusion of GA-based feature selection and the LSTM-RNN architecture significantly amplified the overall effectiveness of the intrusion detection system (IDS).

The evaluation of the IDS necessitated a thorough analysis, focusing on crucial metrics such as accuracy, recall, precision, f-score, and the confusion matrix. This in-depth assessment extensively relied on the NSL-KDD dataset for testing purposes.

Leveraging LSTM-RNN models, the dataset underwent a segmentation process, creating distinct binary classifications (normal and malicious) and multi-class categorizations involving Normal, DoS, Probe, U2R, and R2L categories. This deliberate division facilitated a comprehensive examination of the LSTM-RNN model's performance across diverse intrusion detection scenarios.

What emerged from this exploration were clear improvements in the LSTM-RNN's classification accuracy. These gains were most pronounced when the GA was employed to refine feature selection, showing its impact in enhancing classification accuracy for both binary and multi-class classifications.

m. Intrusion Detection System for NSL-KDD Dataset Using Convolutional Neural Networks (Ding & Zhai, 2018)

This investigation centred around the training of an Intrusion Detection System (IDS) model, utilizing Convolutional Neural Networks (CNN) specifically tailored for the expansive NSL-KDD dataset. The primary objective aimed to explore the intricacies of multi-class classification scenarios, conducting a direct comparison between the CNN-based model and traditional methods such as Random Forest (RF) and Support Vector Machine (SVM). Additionally, the study delved into state-of-the-art deep learning techniques like the Deep Belief Network (DBN) and Long Short-Term Memory (LSTM). The overarching aim was to execute a thorough analysis, shedding light on the unique strengths and limitations inherent in various approaches within the dynamic field of intrusion detection.

The experiment was designed to assess how well the IDS model performed in multi-class classification when stacked against various other methods. Interestingly, the results highlighted that the IDS model outperformed both the traditional machine learning methods and the cutting-edge deep learning methods in this multi-class classification scenario.

During the evaluation of the IDS model utilizing CNN, the attained overall testing accuracy for the five-class classification was 80.1321% in one assessment and 62.3206% in another. These results emphasized the robust capability of the CNN-based IDS model in adeptly classifying intrusions across multiple classes, highlighting its supremacy over both traditional and innovative methods within this context.

2.5 COMPARATIVE ANALYSIS

In this section, we compared several current research of deep learning models, datasets, and accuracy in Intrusion Detection System.

Table 2.8 Comparison of Deep Learning Models in IDS

Citation	Model	Dataset	Classification	Accuracy (Train Dataset)	Accuracy (Test Dataset)	Precision	Recall	F1-Score
(Zhang et al., 2023)	DNN LSTM	NSL-KDD	Binary-Class	98.78% 98.96%	- -	87.1% 88.54%	89.29% 92.86%	92.39% 90.34%
(Behrooz Sezari et al., 2018)	Deep-FFNN	KDD Cup99	Multi-Class	99.86%	-	-	-	-
(Althubiti et al., 2018)	LSTM	CSIC 2010 HTTP	Binary-Class	99.97%	-	99.5%	99.5%	-
(Hsieh & Su, 2021)	DNN	NSL-KDD	Multi-Class	-	79%	-	-	-
(Lokesh Karanam et al., 2020)	CNN-LSTM	NSL-KDD	Multi-Class	99.6%	89.23%	86.86%	-	-
(Fu et al., 2018)	LSTM	NSL-KDD	Multi-Class	97.52%	-	-	-	-
(Ferrag et al., 2020)	DNN RNN CNN	CSE-CIC-IDS2018	Multi-Class	97.28% 97.31% 97.38%	-	-	-	-
(Laghrissi et al., 2021)	LSTM	KDD Cup99	Binary-Class	98.88%	-	-	-	-
(Jiang et al., 2020)	CNN + BiLSTM	NSL-KDD UNSW-NB15	Multi-Class	-	83.58% 77.16%	-	-	-
(Ding & Zhai, 2018)	CNN LSTM	NSL-KDD	Multi-Class	-	80.13% 73.18%	-	-	-

In table 2.8 above, we can see the comparison between each of the researchers' experiment results. We can conclude that most of the Train Dataset used by the researchers and train the accuracy are above 97% and above. But for the Testing Dataset, all the accuracy is below 90% because the testing dataset combined with the real network traffic data.

2.6 CONCLUSION

Deep learning technology spans a broad spectrum of algorithmic models and holds immense potential for research. In this study, we've focused on analyzing the application of deep learning in intrusion detection research over the past five years. The Recurrent Neural Networks (RNNs) stand out due to their proficiency such as Simple-RNN, LSTM and GRU in processing sequential data and capturing temporal dependencies, a crucial capability given the serialized nature of intrusion detection data, such as network traffic or system logs. The adaptability of the RNN model to intrusion detection tasks is noteworthy, and its effectiveness is further emphasized when applied to widely used datasets like NSL-KDD. This dataset's prevalence attests to its reliability, endorsing the suitability of the RNN model for enhancing intrusion detection. In conclusion, the synergy between RNN models and datasets like NSL-KDD presents a promising avenue for advancing our capabilities in detecting and addressing security threats within serialized intrusion detection data. For our experiment results, we trained data similar with other researchers experiment results but for the algorithm & parameter we used different types of units to compare the different testing results using Test Dataset.

CHAPTER III

METHODOLOGY

3.1 INTRODUCTION

The research methodology describes the methods and data analysis used in the study. The relentless growth of the digital landscape, coupled with the expanding reliance on networked systems, has brought network security to the forefront of technological priorities. As organizations and individuals continue to entrust sensitive data to digital environments, safeguarding against network intrusions has become an essential task. To address this challenge, this research endeavours to deploy advanced techniques of deep learning to detect and mitigate network intrusion events.

In this project, the quantitative methodology is strategic and founded on the necessity for a rigorous and data-centric examination of network intrusions. Quantitative methods offer the means to establish a structured and evidence-based foundation for this study, promising robust and replicable results. Deep learning, a subset of machine learning, has emerged as a powerful technique for addressing complex, data-intensive tasks. Its ability to autonomously learn and adapt from data makes it particularly well-suited for the dynamic nature of network intrusion patterns. By using quantitative methods, it can allows not only detect network intrusions, but also to optimize the model's performance through rigorous experimentation.

This chapter is structured to provide a comprehensive account of the research methodology used in our endeavour the performance comparison of deep learning with intrusion detection systems using Recurrent Neural Networks based Framework. It encompasses a range of critical elements, from data collection and preprocessing to model selection, training, and evaluation.

3.2 RESEARCH DESIGN

In this study, we adopt a quantitative research approach. Quantitative research allows for the systematic collection and analysis of numerical data to draw statistical inferences and generalize. This approach is particularly well-suited for examining the effectiveness of deep learning models in network intrusion detection.

This research assumes the form of an experimental study. In this experimental design, we will manipulate independent variables, such as data collection, data preprocessing techniques, model architectures and hyperparameters. While observing their impact on dependent variables, such as detection accuracy, false positives, and false negatives. And the last phases will be evaluation and results. This controlled experimental approach facilitates a meticulous examination of these critical variables. Figure 3.1 discuss the experiment flows (Yee Mon Thant et al., 2023).

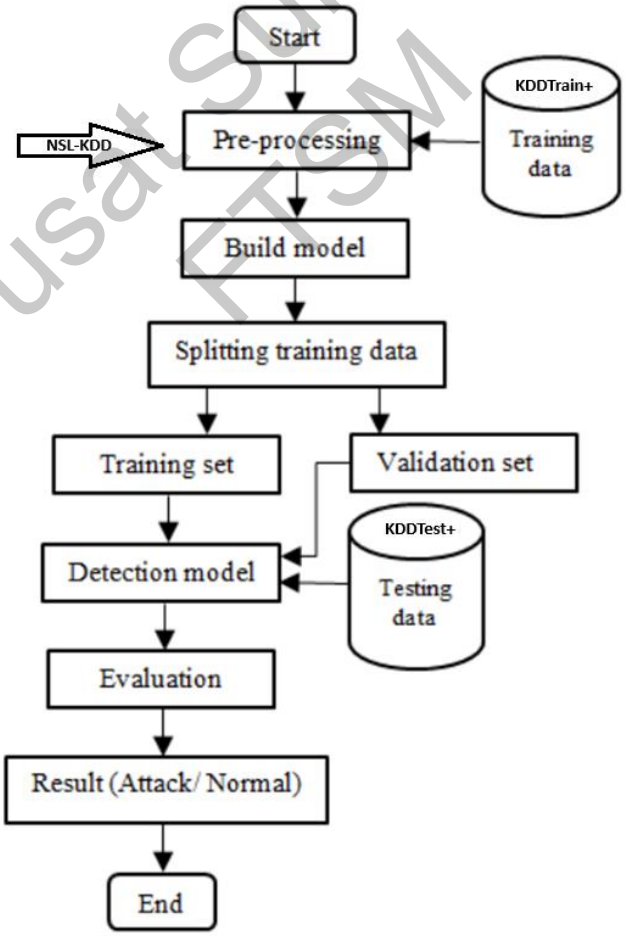


Figure 3.1 Overview of Experiment Flows (Yee Mon Thant et al., 2023)

3.3 DATA COLLECTION

In this project, we've opted to utilize the NSL-KDD dataset as our primary source of network traffic data. Renowned as a benchmark dataset for network intrusion detection, the NSL-KDD dataset was specifically crafted to overcome the limitations of the original KDD Cup 1999 dataset. It provides a more authentic portrayal of network traffic and intrusion scenarios, offering researchers a comprehensive view. This substantial dataset, containing around 4.5 million records, is commonly employed in intrusion detection experiments. In table 3.1 below describe the 41 features are categorized into three primary groups: TCP connection features, content features, and traffic features, enabling a multifaceted analysis of network behavior and potential threats (Maithem & Al-sultany, 2021).

Table 3.1 Features of NSL-KDD Dataset

No	Features Name	Type	Description
		TCP Connection Features	
1	Duration	Numeric	Length (number of seconds) of the connection
2	Protocol Type	Non-numeric	Protocol type tcp, udp and icmp
3	Service	Non-numeric	Network service on the destination, e.g., http, telnet, etc.
4	Source Bytes	Numeric	The number of data bytes transferred from source to destination
5	Destination Bytes	Numeric	The number of data bytes transferred from destination to source
6	Flag	Non-numeric	Normal or error status of the connection
7	Land	Numeric	1 if connection is from/to the same host/port; 0 otherwise
8	Wrong Fragment	Numeric	Number of “wrong” fragments
9	Urgent	Numeric	Number of urgent packets
		Content Features	
10	Hot	Numeric	number of “hot” indicators
11	Number Failed Logins	Numeric	number of failed login attempts
12	Logged In	Numeric	1 if successfully logged in; 0 otherwise
13	Number Compromised	Numeric	number of “compromised” conditions
14	Root Shell	Numeric	1 if root shell is obtained; 0 otherwise

… to be continued

… continuation

15	Su Attempted	Numeric	1 if "su root" command attempted; 0 otherwise
16	Number Root	Numeric	number of "root" accesses
17	Number File Creations	Numeric	number of file creation operations
18	Number Shells	Numeric	number of shell prompts
19	Number Access Files	Numeric	number of operations on access controls files
20	Number Outbound Cmds	Numeric	number of outbound commands in an ftp session
21	Is Host Login	Numeric	1 if the login belongs to the "hot" list; 0 otherwise
22	Is Guest Login	Numeric	1 if the login is a "guest" login; 0 otherwise
	Traffic Features		
23	Count	Numeric	number of connections to the same host as the current connection in the past two seconds
24	Destination Host Count	Numeric	count of the connections having same dst host
25	Serror Rate	Numeric	% of connections that have "SYN" errors
26	Rerror Rate	Numeric	% of connections that have "REJ" errors
27	Same Service Rate	Numeric	% of connections to the same service
28	Different Service Rate	Numeric	% of connections to different services
29	Service Count	Numeric	The number of connections to the same service as the current connection in the previous two seconds
30	Service Serror Rate	Numeric	% of connections that have "SYN" errors
31	Service Rerror Rate	Numeric	% of connections that have "REJ" errors
32	Service Different Host Rate	Numeric	% of connections to different hosts
33	Destination Host Service Count	Numeric	count of connections has same dst host and using same service
34	Destination Host Same Service Rate	Numeric	% of connections have same dst port and using same service
35	Destination Host Different Service Rate	Numeric	% of different services and current host
36	Destination Host Same Source Port Rate	Numeric	% of connection to current host having same src port
37	Destination Host Service Different Host Rate	Numeric	% of connections to same service coming from diff.host
38	Destination Host Serror Rate	Numeric	% of connection to current host that have an S0 error
39	Destination Host Service Serror Rate	Numeric	% of connection to current host and specified service that have an S0 error
40	Destination Host Rerror Rate	Numeric	% of connection to current host that have an RST error
41	Destination Host Service Rerror Rate	Numeric	% of connection to the current host and specified service that have an RST error

3.3.1 Multiclassification Attacks

There are 22 kinds of attacks in this dataset, which can be grouped into 4 major categories:

1. A Denial-of-Service Attack (DoS) happens when an attacker blocks legitimate users' access to a system by overwhelming system resources, such as computing power or memory to the extent that it becomes incapable of managing genuine requests. For instance, a SYN flood is a typical example where an excess of SYN requests floods the system, disrupting its ability to attend to valid user demands.
2. A user-to-root attack (U2R) occurs when an attacker, having local access as a regular user on the system (possibly acquired through methods like dictionary attacks, password sniffing, or social engineering) exploits system vulnerabilities to escalate their privileges to gain root access. This elevated access grants them the capabilities of a system supervisor. An example of this type of attack includes different forms of "buffer overflow" attacks targeting vulnerabilities within the system.
3. In a Remote to Local Attack (R2L), the attacker initiates the attack by transmitting packets from a remote machine across a network, attempting to gain unauthorized access without legitimate credentials. This unauthorized access might be achieved through methods like password guessing. The attacker seeks to exploit vulnerabilities from a distance, attempting to breach the targeted system's defenses without having any prior authorization.
4. Probing, also known as a Probe attack, involves an attacker attempting to gather information about a network to identify potential vulnerabilities. Through these investigative actions, the attacker aims to map the network's topology and uncover the array of services active within it. For instance, techniques like port scanning are utilized to scrutinize the network, allowing the attacker to pinpoint possible weaknesses and understand the types of services operating on the network.

The NSL-KDD dataset comprises two main components: KDDTrain+ serves as a comprehensive training dataset, available in both CSV and txt formats, containing attack-type labels and difficulty levels. Similarly, KDDTest+ functions as a thorough

testing dataset, also provided in CSV and txt formats, and includes attack-type labels alongside difficulty levels. These datasets collectively offer a detailed repository for training and evaluating intrusion detection models, enabling researchers to work with labeled data for comprehensive analysis and testing (Muhuri et al., 2020). Table 3.2 and 3.3 characterize the NSL-KDD dataset.

Table 3.2 Categories of Attacks

Major Categories	Subcategories
Denial of Service (DoS)	ping of Death, LAND, neptune, backscatter, smurf, teardrop
User to Root (U2R)	buffer Overflow, loadmodule, perl, rookit
Remote to Loval (R2L)	ftp-write, password guessing, imap, multi-hop, phf, spy, warezclient, warezmaster
Probing	ipsweeping, nmap, postsweeping, satan

Table 3.3 Total instances by attack type in the NSL-KDD Dataset-

Attack	Total Instances in NSL-KDD Dataset	Attack Category
Back	956	DoS
Land	18	
Neptune	41,214	
Pod	201	
Smurf	2646	
Teardrop	892	
Satan	3633	
Ipsweep	3599	Probe
Nmap	1493	
Portsweep	2931	
Normal	67,343	Normal
guess-passwd	53	R2L
ftp-write	8	
Imap	11	
Phf	4	
Multihop	7	
Warezmaster	20	
Warezclient	1020	
Spy	2	
buffer-overflow	30	U2R
Loadmodule	9	
Perl	3	
Rootkit	2931	

3.4 DATA PREPROCESSING

The NSL-KDD dataset stands out for its clarity, devoid of any noise or missing values, ensuring a clean and comprehensive dataset for analysis. However, it presents a challenge with a mix of numerical and text values. The numerical values, particularly the presence of large numbers, pose potential delays in training and add complexity to the processing of the dataset due to the scale and magnitude of these numerical entries.

3.4.1 Numericalization

In the NSL-KDD dataset, among its 41 features, 38 are numeric while 3 are non-numeric, including 'protocol_type', 'service', and 'flag'. To fit these into neural network architectures, non-numeric attributes undergo a transformation into numeric forms. Take 'protocol_type', for instance, where 'tcp', 'udp', and 'icmp' get encoded into binary vectors (1,0,0), (0,1,0), and (0,0,1) respectively, utilizing techniques like the One-Hot-Encoder to ensure their compatibility with numerical processing. This transformation results in a sparse matrix, where each column represents a possible value for a particular feature. Similarly, 'service' encompasses 70 attributes, while 'flag' involves 11 attributes. Employing this conversion approach, the original 41-dimensional feature set expands to a 122-dimensional space (comprising 38 numeric, 3 non-numeric, 70 'service', and 11 'flag' attributes) after the transformation process. (Muhuri et al., 2020). Figure 3.2 explain the one hot encoder (Maithem & Al-sultany, 2021) and figure 3.3 is the features for protocol type, service, and flag (Saporito, 2019).

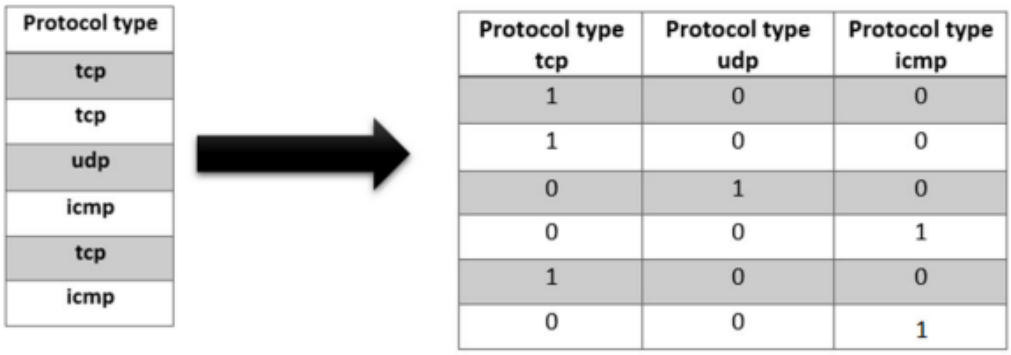


Figure 3.2 One hot Encoder on Protocol Type Column (Maithem & Al-sultany, 2021)

Protocol Type (2)	Service (3)				Flag (4)
• icmp • tcp • udp	• other • link • netbios_ssn • smtp • netstat • ctf • ntp_u • harvest • efs • klogin • systat • exec • nntp • pop_3 • printer • vmnet • netbios_ns	• urh_i • ssh • http_8001 • iso_tsap • aol • sql_net • shell • supdup • auth • whois • discard • sunrpc • urp_i • Rje • ftp • daytime • domain_u • pm_dump	• time • hostnames • name • ecr_i • bgp • telnet • domain • ftp_data • nnsp • courier • finger • uucp_path • X11 • imap4 • mtp • login • tftp_u • kshell	• private • http_2784 • echo • http • ldap • tim_i • netbios_dgm • uucp • eco_i • Remote_job • IRC • http_443 • red_i • Z39_50 • Pop_2 • gopher • Csnet_ns	• OTH • S1 • S2 • RSTO • RSTRs • RSTOS0 • SF • SH • REJ • S0 • S3

Figure 3.3 Features for Protocol Type, Service and Flag (Saporito, 2019)

3.4.2 Normalization

Certain attributes within the dataset, such as 'duration [0,58329]', 'src_bytes [0,1.3 × 10^9]', and 'dst_bytes [0,1.3 × 10^9]', showcase a notable difference between their maximum and minimum values. To address this variation, a logarithmic scaling technique is employed, transforming these ranges into more manageable scales: 'duration [0,4.77]', 'src_bytes [0,9.11]', and 'dst_bytes [0,9.11]'. Following this, all feature values are standardized within the [0,1] range using a linear mapping approach based on specific feature maximums and minimums (as denoted by the formula (1)). This scaling method ensures consistency and prepares these attributes for seamless integration into the neural network model (Yin et al., 2017).

$$x_i = \frac{x_i - Min}{Max - Min} \qquad (1)$$

3.5 DEEP LEARNING MODELLING

Choosing the most suitable deep learning models for network intrusion detection is a pivotal decision, one that carries significant implications for the overall effectiveness of our intrusion detection system. In this section, we'll delve into the criteria we meticulously applied to select the specific deep learning models for our research and the rationale behind these choices. Our decisions were grounded in a thorough evaluation of factors encompassing model performance, insights from previous research, and the compatibility of these models with the intricacies of our NSL-KDD dataset.

3.5.1 Criteria For Model Selection

In this project choice of deep learning models hinged on several critical criteria, reflecting the dedication to a robust and effective network intrusion detection system:

a. Model Performance: Paramount among our criteria was a model's proven performance in the demanding realm of intrusion detection. We sought models that consistently demonstrated their prowess in achieving high detection accuracy, precision, recall, and F1-score.
b. Previous Research: A diligent review of existing literature served as a guiding light in our model selection process. We were keen to build upon the knowledge and successes found in previous research studies. Models that had garnered recognition for their efficacy and resilience in intrusion detection experiments were given precedence.
c. Suitability to Dataset: The alignment of deep learning models with the unique attributes of the NSL-KDD dataset was another pivotal factor in our decision-making. It was imperative that the chosen models could seamlessly navigate the dataset's dimensions, class distribution, and the diverse nature of network traffic data.

3.5.2 Model Selection

In this project, we have selected the following deep learning RNN models as the keystones of our network intrusion detection research:

a) Recurrent Neural Networks (Simple-RNN):

- RNNs, designed for sequential data analysis, are perfectly attuned to the temporal nature of network traffic patterns. Their capability to decipher evolving sequences of events in network data positions them as a robust choice for intrusion detection. Simple RNNs form the backbone of sequential data processing, but their struggle with retaining long-term information limits their effectiveness in modelling dependencies across lengthy sequences (Shiri et al., 2023).

b) Long Short-Term Memory Networks (LSTMs):

- LSTMs, a specialized variant of RNNs, shine in capturing long-range dependencies within sequential data. In our context, they offer the potential to decipher intricate, time-delayed intrusion behaviors, thereby enhancing the detection process. LSTM tackles the limitations of traditional RNNs by introducing a complex memory cell mechanism, allowing for the retention and selective management of information over extended sequences through its input, forget, and output gates (Shiri et al., 2023).

c) Gated Recurrent Unit (GRU):

- GRU simplifies the LSTM architecture by amalgamating the cell and hidden states, streamlining the gates into a single "update gate," offering computational efficiency without compromising its ability to capture long-term dependencies (Shiri et al., 2023).

The deep learning models chosen are a manifestation of a meticulous selection process, woven together by considerations of model performance, insights from previous research, and the dataset's unique attributes. Through this approach, we aim to undertake a comprehensive exploration of their capabilities and determine the most potent models for network intrusion detection within our research context. The selection process underlines our commitment to best practices and the relentless pursuit of excellence in this dynamic field.

3.6 MODEL TRAINING

Model training represents a critical phase in our research, where we harness the power of deep learning to equip our models for network intrusion detection. In this section, we elaborate on our approach to training these models, detailing the selection of loss functions, optimization algorithms, hyperparameter tuning, and the pivotal role of validation and test sets in evaluating model performance.

1. Lost Functions
2. Optimization Algorithms
3. Hyperparameter Tuning
4. Validation and Test Set

3.7 MODEL DETECTION

Within our methodology chapter, the Detection section stands as a pivotal aspect of our research pursuit. Here, we delve into the complexities of implementing our meticulously chosen deep learning models for network intrusion detection. Our array of deep learning models, encompassing distinct architectures like Simple-RNN, Long Short-Term Memory Networks (LSTMs), and gated recurrent unit (GRU), will be actively employed to address the challenge of intrusion detection. These models have undergone rigorous training, ensuring they are equipped with optimized loss functions, and finely tuned hyperparameters, all calibrated for peak performance. We'll gauge the efficiency of the intrusion detection system by employing a range of performance metrics like accuracy, precision, recall, F1-score, and area under the ROC curve. Our evaluation aims to measure the capability of the deep learning models, whether used independently or in combinations, in effectively recognizing network intrusions while striving to reduce false alarms.

3.8 MODEL EVALUATION

The Evaluation section of our methodology is pivotal, as it delves into the processes and metrics, we employ to assess the performance of our deep learning models in the realm of network intrusion detection. This section provides a comprehensive overview of our approach to evaluating model effectiveness.

To gauge the performance of our deep learning models, we rely on a suite of performance metrics, each offering unique insights into the model's behaviour. The key metrics we employ include:

a. **Accuracy**: This metric assesses the general accuracy of our model's predictions, offering an insight into its capability to precisely classify both normal and intrusive network traffic.
b. **Precision**: Precision measures the accuracy of positive predictions among all predictions labeled as positive. It's a crucial gauge of the model's capacity to minimize false alarms.
c. **Recall** (Sensitivity): Recall measures the proportion of true positives out of all actual positives. It indicates the model's effectiveness in capturing instances of network intrusion.
d. **F1-Score**: The F1-score serves as a comprehensive metric, striking a balance between precision and recall, providing a consolidated measure of the model's overall performance.
e. **Area under the ROC Curve (AUC-ROC)**: AUC-ROC provides insights into the model's ability to discriminate between normal and intrusive traffic, particularly in scenarios where the class distribution is imbalanced.

In our study, we assessed the efficacy of the proposed Simple-RNN, LSTM, and GRU models using various performance metrics: Accuracy (2), Precision (3), Recall (4), and F1-score (5).

Accuracy illustrates the proportion of accurately predicted instances compared to the total predictions made.

$$Accuracy = \frac{Tp+Tn}{Tp+Tn+Fp+Fn} \qquad (2)$$

Precision reveals the ratio of correctly identified malicious packets to the entire set predicted as malicious.

$$Precision = \frac{Tp}{Tp+Fp} \qquad (3)$$

Recall quantifies the fraction of actual malicious packets correctly identified by the model.

$$Recall = \frac{Tp}{Tp+Fn} \qquad (4)$$

F1 score computes the harmonic mean of precision and recall, offering a comprehensive metric to evaluate the overall classification performance.

$$F1 - Score = \frac{2\times(Precision\times Recall)}{Precision+Recall} \qquad (5)$$

The formula above outlines the specific details of these parameters: TP (True Positive), TN (True Negative), FP (False Positive), and FN (False Negative) (G.Janani Pandeeswari & S. Jeyanthi, 2022).

3.9 RESULT

To evaluate the relative performance of different deep learning models, we engage in model comparison. This involves a meticulous assessment of each model's performance using the same evaluation metrics, enabling us to identify the model or ensemble that excels in detecting network intrusions. At the end of summary, when presented in the main findings or results chapter, will offer a comprehensive account of our research outcomes. By presenting a well-organized and visually appealing overview of the data, the performance of deep learning models, comparisons, real-world testing insights, and ethical considerations, we aim to provide a holistic understanding of the effectiveness of our network intrusion detection system.

CHAPTER IV

EXPERIMENT AND ANALYSIS

4.1 INTRODUCTION

In this section, our project presented the overall of the framework for network intrusion detection and introduced the RNN-IDS for 3 types of architecture. We aim to assess and evaluate the performance of our model using a comprehensive set of experiments conducted on the network intrusion detection dataset (NSL-KDD). Moreover, we presented the time of training each of the models which is Simple-RNN, RNN-LSTM and RNN-GRU. Each of the models displayed 3 different units of parameter, which is 64, 128 and 256 units. At the end of the experiments, we employ various metrics such as Accuracy, Precision, Recall and F1-score. Accuracy for train-test set, accuracy for KDD Testing dataset, and evaluations demonstrated our model's advantages and effectiveness compared to other papers.

4.2 EXPERIMENT ENVIRONMENT

In this section, we describe the hardware and software used in carrying out the experiments in this project.

4.2.1 Hardware

The experiments in this project were carried out on a hardware system running Windows 11 Pro Version (22H2). The system has a CPU 12th Gen Intel® Core ™ i7-12700H 2.70 GHz base speed processor, 40GB of RAM, 1TB Solid-State Drive running the operating system, and NVIDIA Geforce RTX3050 for GPU.

4.2.2 Software

In these experiments, Python was the main programming language used for the experiments. Anaconda was installed for create an environment and allows us to install all the libraries and framework for deep learning. Anaconda Jupyter Notebook used to create interactive notebook documents that can contain live code, equations, visualizations, media, and other computational outputs. The deep learning framework utilized as Keras, an open-source neural network library written in Python, we integrated combination with Keras with the popular backed libraries such as TensorFlow. For the normalizing data, metric calculations, dataset for train and testing, we integrated the Sklearn to run all these experiments. Lastly, any third-party libraries such as plotting the graph and load the dataset, reshape the data, we used libraries such as Seaborn and Matplotlib, NumPy and Pandas.

4.3 EXPERIMENTAL DESIGN

In this experiment, we proposed three types of the model, which is Simple-RNN, LSTM and GRU, figure 4.1 (Kasongo, 2022), comparative analysis to the 2-class and 5-class experiments. Each of these models we conducted different units of parameter to classify instances in the NSL-KDD intrusion dataset. The 2-class classification are normal or malicious. Besides that, the 5-class classification are identifying specific types of intrusions, namely DoS, Probe, R2L, U2R and normal. At the end of the experiments, we compared to the existing model, which is DNN (Liu et al. 2020), and perform each of the RNN models analysis, accuracy, and time to choose the most valuable deep learning detection for network intrusion.

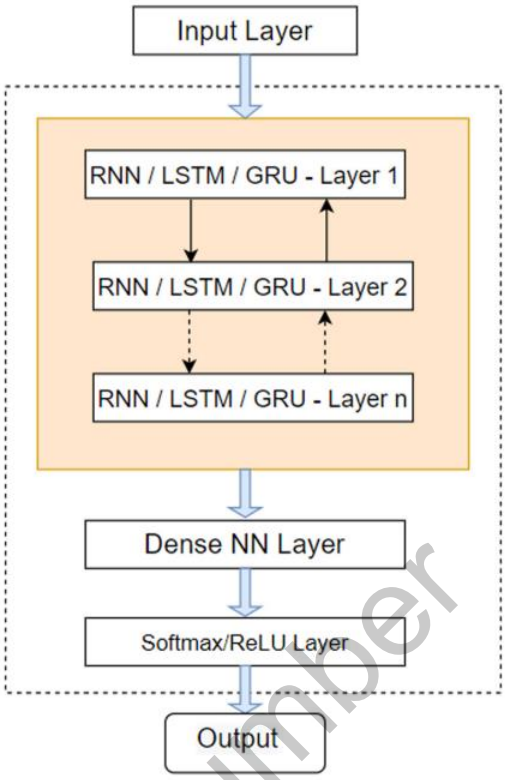


Figure 4.1 RNN/LSTM/GRU Structure (Kasongo, 2022)

4.3.1 Split The Dataset

Access to robust NSL-KDD datasets plays a pivotal role in the development and evaluation of efficient IDS algorithms. These datasets serve as critical tools for researchers and practitioners, enabling comprehensive testing of algorithms across diverse attack scenarios. By utilizing IDS datasets, valuable patterns and trends in network attacks can be identified, offering insightful perspectives for bolstering overall network security (Vanlalruata Hnamte & Hussain, 2023). In this experimental setup, two distinct datasets were employed. The initial training of the model utilized the KDDTrain+.txt dataset, while the evaluation phase involved the KDDTest+.txt dataset. Further evaluation was performed by splitting the original training set into two segments: one for training the model and the other for testing its performance, more info discussed in section 4.3.3. This meticulous approach focused on assessing the intrusion detection model, emphasizing high accuracy and a minimal false alarm rate. A detailed breakdown of attack classifications into 2-class and 5-class categories is illustrated starting from the tables 4.1, 4.2, 4.3 and 4.4. Figures 4.2, 4.3, 4.4 and 4.5 are the pie chart for each of the train and test dataset attacks (Lokesh Karanam et al., 2020).

a. KDDTrain+ Dataset

i. 2-class classification (binary-class)

Table 4.1 Binary-Class for Train Dataset

Attack	Total	Percentage	Intrusion (Label)
Normal	67343	53.46%	1
Malicious	58630	46.54%	0

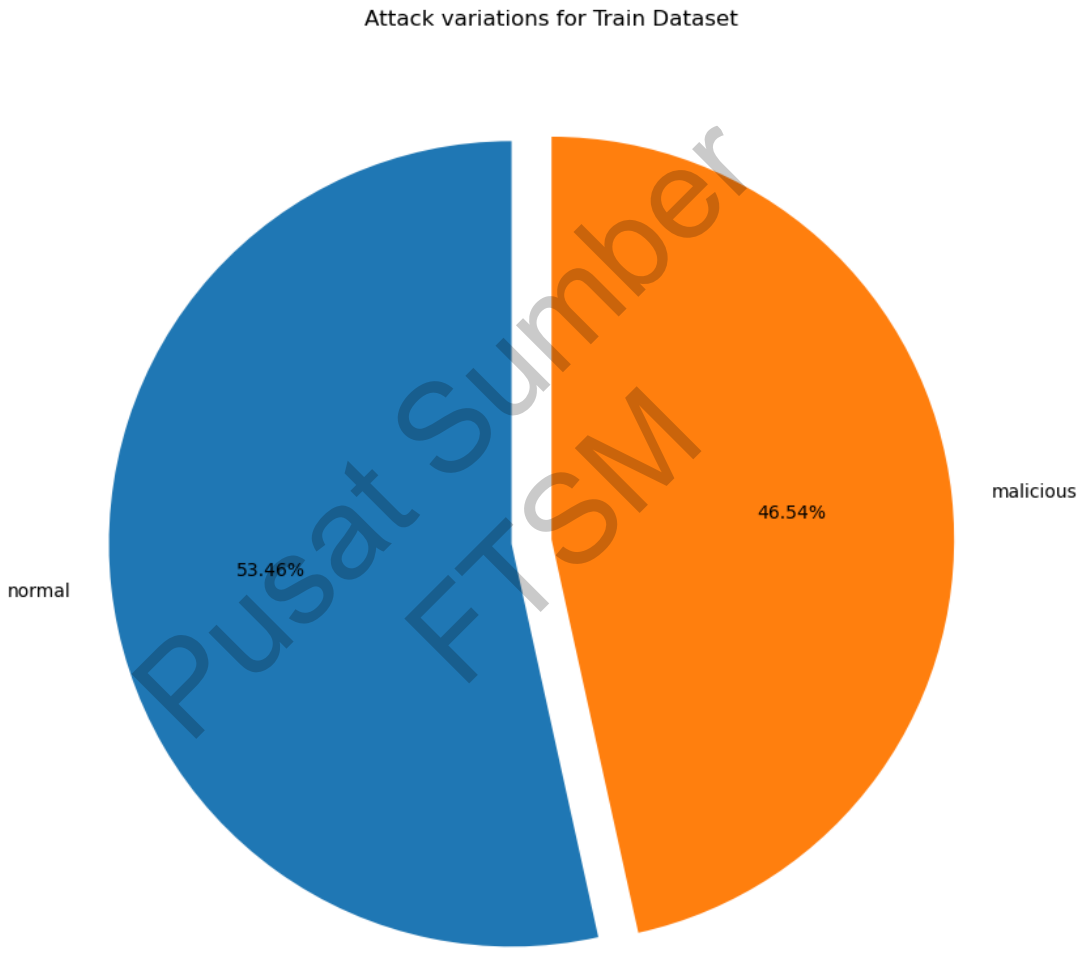


Figure 4.2 Pie Chart for Binary-Class Train Dataset

ii. 5-class classification (multi-class)

Table 4.2 Multi-Class for Train Dataset

Attack	Total	Percentage	Intrusion (Label)
DoS	45927	36.46%	0
Probe	11656	9.25%	1
R2L	995	0.79%	2
U2R	52	0.04%	3
Normal	67343	53.46%	4

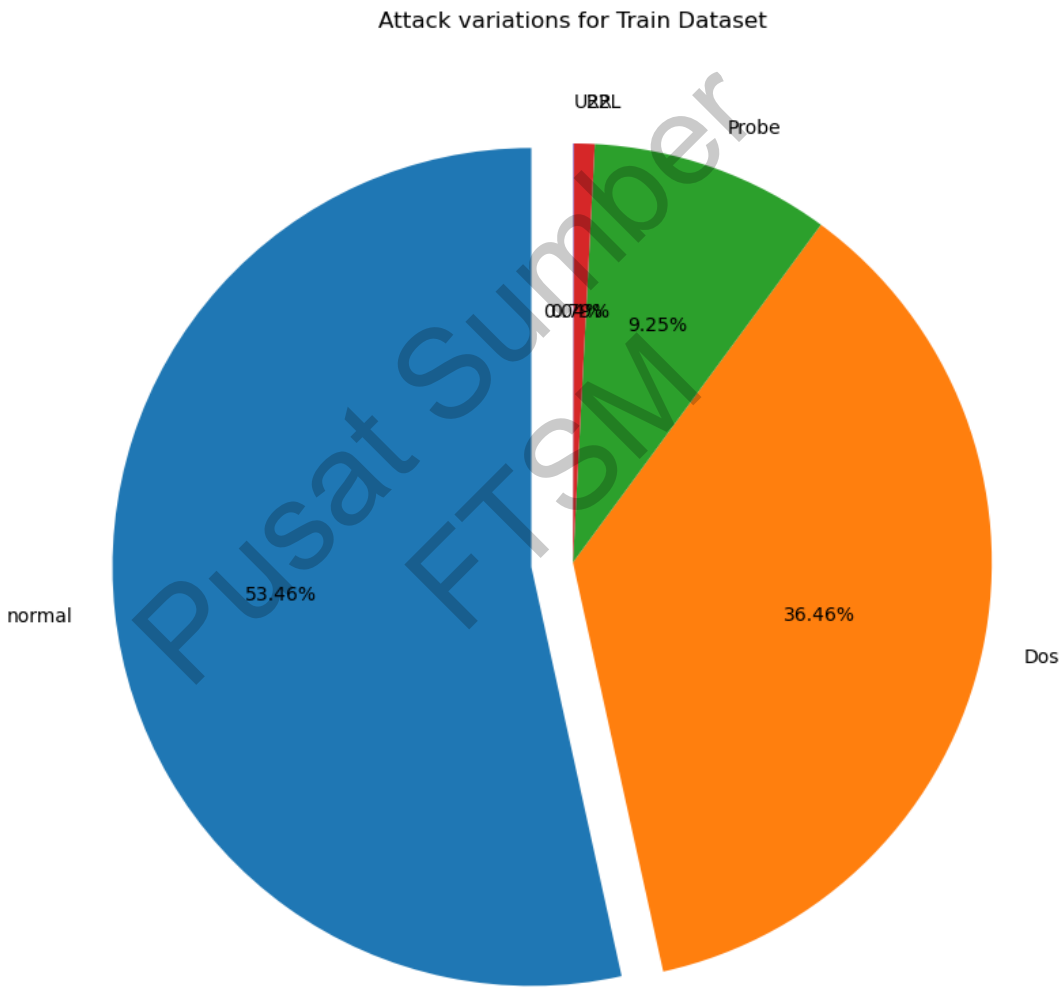


Figure 4.3 Pie Chart for Multi-Class Train Dataset

b. KDDTest+ Dataset

i. 2-class classification (binary-class)

Table 4.3 Binary-Class for Test Dataset

Attack	Total	Percentage	Intrusion (Label)
Normal	9711	43.08%	1
Malicious	12833	56.92%	0

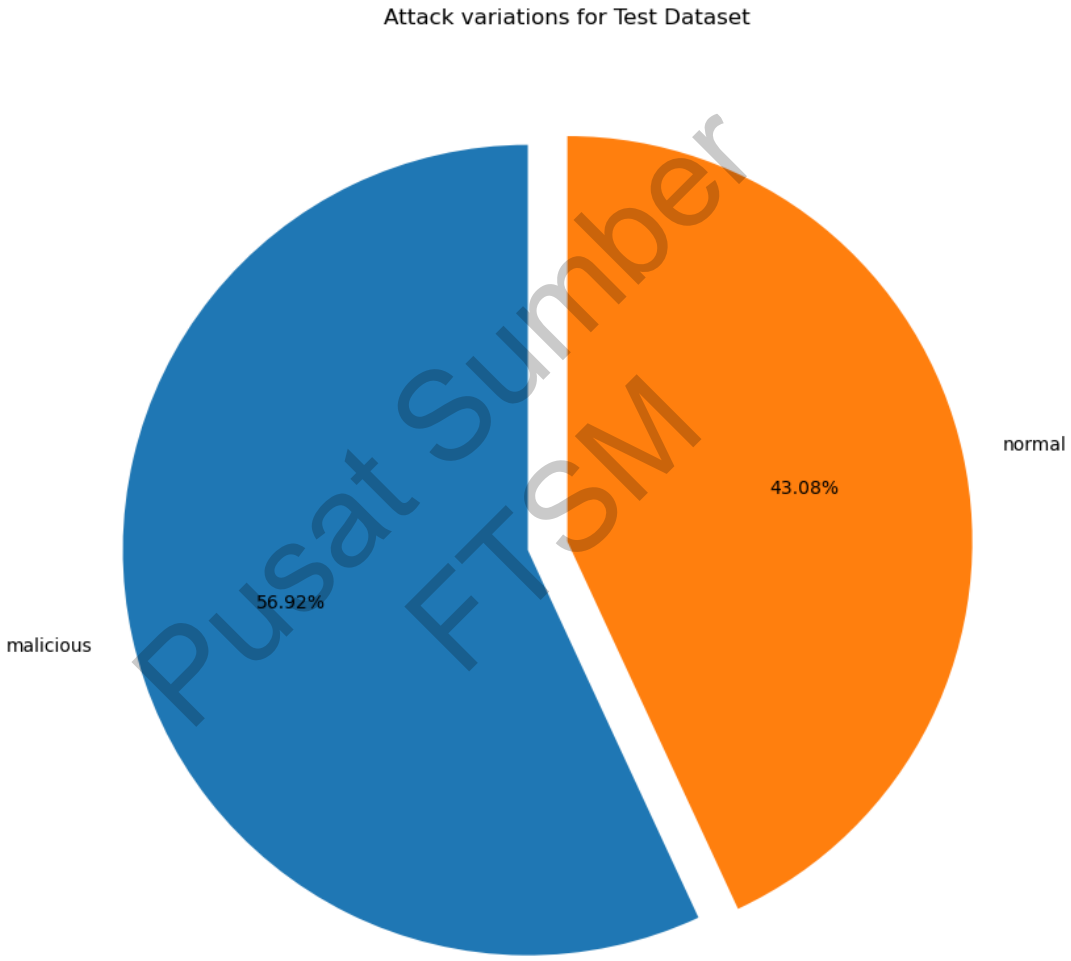


Figure 4.4 Pie Chart for Binary-Class Test Dataset

ii. 5-class classification (multi-class)

Table 4.4 Multi-Class for Test Dataset

Attack	Total	Percentage	Intrusion (Label)
DoS	7460	33.09%	0
Probe	2421	10.74%	1
R2L	2885	12.80%	2
U2R	67	0.30%	3
Normal	9711	43.08%	4

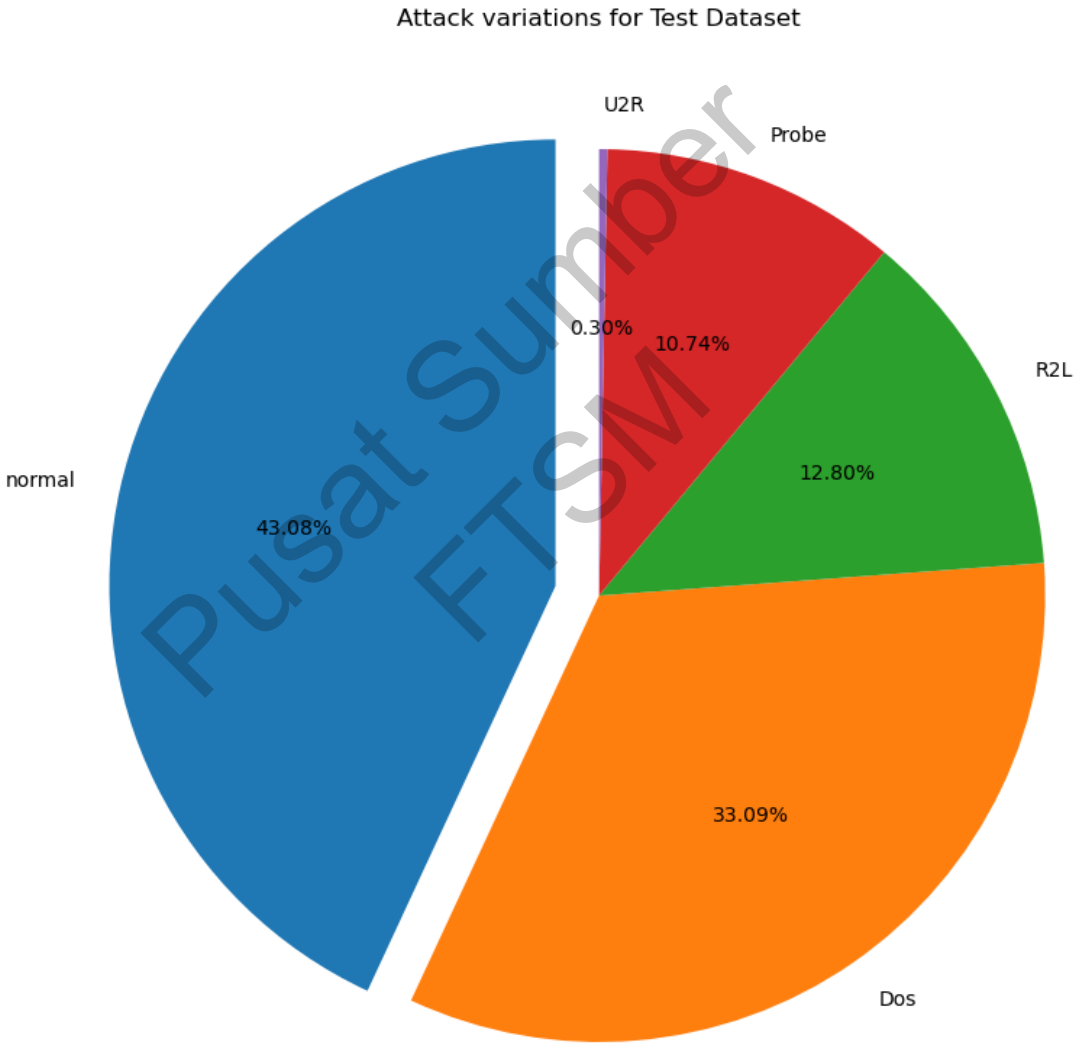


Figure 4.5 Pie Chart for Multi-Class Test Dataset

4.3.2 Data Preprocessing

Data preprocessing plays a pivotal role in preparing datasets for machine learning by transforming them into an appropriate format. This procedure encompasses multiple steps, starting with assessing the data quality, followed by cleansing the dataset by eliminating any corrupt or erroneous data entries. The subsequent phases involve data transformation and ultimately data reduction to streamline the dataset for analysis. In our approach, we applied One-hot-encoding specifically to categorize the columns related to protocol_type, service, and flag, facilitating efficient handling of categorical data within the dataset. In the figure 4.6 below, we demonstrate for one-hot-encoding.

```
# one-hot-encoding categorical columns
data = pd.get_dummies(data,columns=['protocol_type','service','flag'],prefix="",prefix_sep="")
print(data.shape)
```

```
(125973, 123)
```

```
# one-hot-encoding categorical columns
data_test = pd.get_dummies(data_test,columns=['protocol_type','service','flag'],prefix="",prefix_sep="")
print(data_test.shape)
```

```
(22544, 117)
```

Figure 4.6 One-Hot-Encoding

4.3.3 Data Separation

The dataset comprises an extensive volume of data, necessitating a focus on high-quality data for this project's objectives. To ensure this, the data is divided into two distinct categories: the training dataset and the testing dataset. Within this project's framework, 80% of the training dataset is allocated for training purposes, while the remaining 20% is reserved for testing and development phases. This Train-Test Split methodology serves the purpose of preventing data overfitting, ensuring that the

model's performance extends beyond just the training data, ultimately enhancing its generalizability. Figure 4.7 below shows the train-test split to 80% and 20%.

```
# Split Train dataset into train and validation sets
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.2, random_state=42)
print(x_train.shape,'\n',x_test.shape)
print(y_train.shape,'\n',y_test.shape)
```

```
(100778, 122)
 (25195, 122)
(100778, 1)
 (25195, 1)
```

Figure 4.7 Train-Test Split

4.3.4 Reshape the Train and Test Dataset

The code modifies the structure of the input data to fit a specific format commonly used in machine learning tasks, particularly for analyzing sequences or time-series data. In the figure 4.8, it reshapes the x_train and x_test datasets into a 3D shape of [samples, time steps, features]. This format, typical for sequences:

a. Defines **samples** as the number of observations.
b. Represents **time** steps as the sequence length.
c. Depicts **features** as the number of variables per time step.


```
# reshape input to be [samples, time steps, features]
x_train = np.reshape(x_train, ( x_train.shape[0], 1 , x_train.shape[1] ))
x_test = np.reshape(x_test, ( x_test.shape[0], 1,  x_test.shape[1] ))
```

```
xl_data_test_reshaped = np.reshape(xl_data, (xl_data.shape[0], 1, xl_data.shape[1]))
```

Figure 4.8 Reshape the Crucial

4.4 HYPERPARAMETER TUNING

Hyperparameters represent predefined settings established before training begins, distinct from parameters learned from data during the training process. It's exerted influence over various aspects of the learning algorithm's behaviour, including the model's complexity, learning rate, and its ability to discern patterns within the dataset.

4.4.1 Parameters

We used the 3 hidden layers to train our RNN models, with the following parameters which is (64, 64, 64), (128, 128, 128), and (256, 256, 256) used in our intrusion detection models: Simple-RNN, LSTM, and GRU. We employed the sigmoid and softmax activation functions and implemented the Adam optimizer during our experimental setup. We used binary_crossentropy and categorical_crossentropy for loss fuction. Sigmoid & binary_crossentropy for 2-class classification and softmax & categorical_crossentropy for 5-class classification.

4.4.2 Sigmoid

The sigmoid function, labelled as σ(z), serves as an activation function that transforms real numbers into values within the 0 to 1 range. This mathematical operation, defined by formula (6), finds extensive use in binary classification scenarios by computing probabilities. By compressing input values, the sigmoid function readies them for tasks involving binary categorization. Nonetheless, it comes with a drawback: when 'z' reaches extremely high or low values, the sigmoid function triggers vanishing gradients. This issue, where the gradient becomes exceedingly minute, contributes to a slowdown in training deep neural networks. (Furnieles, 2022).

$$(z) = \frac{1}{1+e^{-z}} \qquad (6)$$

4.4.3 Softmax

The softmax function use to classify the output for multiclass classification responsibilities. Its primary function involves taking a set of varied real-valued scores and transforming them into probabilities, ensuring that their collective sum amounts to 1. Formula (7) delineates the mechanics of the softmax function, where 'z' signifies the input vector, while 'K' stands for the count of classes involved. Particularly beneficial in scenarios encompassing multiple classes, softmax shines by generating a comprehensive probability distribution across all available classes. Its widespread adoption extends to neural networks, where it frequently assumes the role of the output

layer activation function to address the complexities of multiclass classification (Furnieles, 2022).

$$(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \qquad (7)$$

4.4.4 Dropout Layer

Dropout is a regularization technique used in neural networks to prevent overfitting. During training, random neurons (nodes) in the network are ignored or "dropped out" with a certain probability. This prevents any one neuron from becoming too influential in the network, forcing the network to learn more robust features. It essentially helps in creating a more generalized model (Chollet, 2018).

4.4.5 Dense Layer

A dense layer is a fully connected layer in a neural network. Each neuron in a dense layer receives input from every neuron of the previous layer, hence forming a dense connection between them. These layers perform a linear operation on the input data followed by a non-linear activation function, allowing the network to learn complex patterns in the data (Chollet, 2018).

4.4.6 Flatten Layer

The flatten layer is used to convert multidimensional data into a one-dimensional array. In neural networks, especially when transitioning from convolutional layers (which work with 3D data like images) to fully connected dense layers (which expect 1D input), the flatten layer reshapes the data without altering its content. For instance, in image processing, it takes a 2D array (image) and transforms it into a 1D array to be fed into a dense layer (Chollet, 2018).

4.5 ANALYSIS OF EXPERIMENT RESULTS

In this project, we performed six sets RNN model of experiments, namely Simple-RNN-2-class, LSTM-2-class, GRU-2-class, Simple-RNN-5-class, LSTM-5-class, and GRU-5-class. For these six sets experiments, we tested for different type units of parameter and classification, at the end we will have 18 experiment results. All these experiments used the official training dataset and evaluate the performance of the models on the testing dataset, it's essential to considered metrics for Accuracy, Precision, Recall, F1-score, and AUC-ROC. In our experiment, we used 128 batch size and 50 epochs to train each of the model performance.

4.5.1 Experiment Architecture

In figure 4.9 below, there is a diagram for the whole of the experiments we used. There are 3 different inputs layer from the start, which is RNN, LSTM, and GRU. In the middle of the diagram, there are 3 hidden layers of the model, and each of the experiment parameters is (64, 64, 64), (128, 128, 128) and (256, 256, 256). In the section below, we provided the model summary 2-class & 5 class for each of the model.

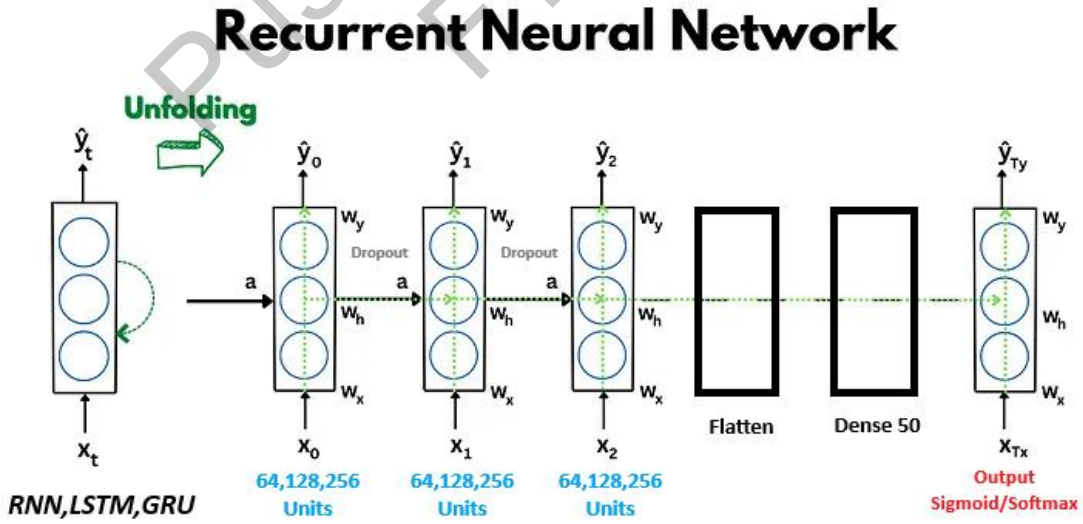


Figure 4.9 RNN Diagram for the Experiments

4.5.2 2-Class of Model Summary

In the figure 4.10, 4.11 and 4.12, these are the model summary for RNN, LSTM and GRU for 2-class 64 units parameters.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 1, 64)             11968

 dropout (Dropout)           (None, 1, 64)             0

 simple_rnn_1 (SimpleRNN)    (None, 1, 64)             8256

 dropout_1 (Dropout)         (None, 1, 64)             0

 simple_rnn_2 (SimpleRNN)    (None, 1, 64)             8256

 flatten (Flatten)           (None, 64)                0

 dense (Dense)               (None, 50)                3250

 dense_1 (Dense)             (None, 1)                 51

=================================================================
Total params: 31781 (124.14 KB)
Trainable params: 31781 (124.14 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 4.10 RNN 2-Class 64 Units

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 1, 64)             47872

 dropout_2 (Dropout)         (None, 1, 64)             0

 lstm_1 (LSTM)               (None, 1, 64)             33024

 dropout_3 (Dropout)         (None, 1, 64)             0

 lstm_2 (LSTM)               (None, 1, 64)             33024

 flatten_1 (Flatten)         (None, 64)                0

 dense_2 (Dense)             (None, 50)                3250

 dense_3 (Dense)             (None, 1)                 51

=================================================================
Total params: 117221 (457.89 KB)
Trainable params: 117221 (457.89 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 4.11 LSTM 2-Class 64 Units

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 1, 64)             36096

 dropout_4 (Dropout)         (None, 1, 64)             0

 gru_1 (GRU)                 (None, 1, 64)             24960

 dropout_5 (Dropout)         (None, 1, 64)             0

 gru_2 (GRU)                 (None, 1, 64)             24960

 flatten_2 (Flatten)         (None, 64)                0

 dense_4 (Dense)             (None, 50)                3250

 dense_5 (Dense)             (None, 1)                 51

=================================================================
Total params: 89317 (348.89 KB)
Trainable params: 89317 (348.89 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 4.12 GRU 2-Class 64 Units

In the figure 4.13, 4.14 and 4.15, these are the model summary for RNN, LSTM and GRU for 2-class 128 units parameters.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 1, 128)            32128

 dropout (Dropout)           (None, 1, 128)            0

 simple_rnn_1 (SimpleRNN)    (None, 1, 128)            32896

 dropout_1 (Dropout)         (None, 1, 128)            0

 simple_rnn_2 (SimpleRNN)    (None, 1, 128)            32896

 flatten (Flatten)           (None, 128)               0

 dense (Dense)               (None, 50)                6450

 dense_1 (Dense)             (None, 1)                 51

=================================================================
Total params: 104421 (407.89 KB)
Trainable params: 104421 (407.89 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 4.13 RNN 2-Class 128 Units

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 1, 128)            128512

 dropout_2 (Dropout)         (None, 1, 128)            0

 lstm_1 (LSTM)               (None, 1, 128)            131584

 dropout_3 (Dropout)         (None, 1, 128)            0

 lstm_2 (LSTM)               (None, 1, 128)            131584

 flatten_1 (Flatten)         (None, 128)               0

 dense_2 (Dense)             (None, 50)                6450

 dense_3 (Dense)             (None, 1)                 51

=================================================================
Total params: 398181 (1.52 MB)
Trainable params: 398181 (1.52 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 4.14 LSTM 2-Class 128 Units

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 1, 128)            96768

 dropout_4 (Dropout)         (None, 1, 128)            0

 gru_1 (GRU)                 (None, 1, 128)            99072

 dropout_5 (Dropout)         (None, 1, 128)            0

 gru_2 (GRU)                 (None, 1, 128)            99072

 flatten_2 (Flatten)         (None, 128)               0

 dense_4 (Dense)             (None, 50)                6450

 dense_5 (Dense)             (None, 1)                 51

=================================================================
Total params: 301413 (1.15 MB)
Trainable params: 301413 (1.15 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 4.15 GRU 2-Class 128 Units

In the figure 4.16, 4.17 and 4.18, these are the model summary for RNN, LSTM and GRU for 2-class 256 units parameters.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 1, 256)            97024

 dropout (Dropout)           (None, 1, 256)            0

 simple_rnn_1 (SimpleRNN)    (None, 1, 256)            131328

 dropout_1 (Dropout)         (None, 1, 256)            0

 simple_rnn_2 (SimpleRNN)    (None, 1, 256)            131328

 flatten (Flatten)           (None, 256)               0

 dense (Dense)               (None, 50)                12850

 dense_1 (Dense)             (None, 1)                 51

=================================================================
Total params: 372581 (1.42 MB)
Trainable params: 372581 (1.42 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 4.16 RNN 2-Class 256 Units

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 1, 256)            388096

 dropout_2 (Dropout)         (None, 1, 256)            0

 lstm_1 (LSTM)               (None, 1, 256)            525312

 dropout_3 (Dropout)         (None, 1, 256)            0

 lstm_2 (LSTM)               (None, 1, 256)            525312

 flatten_1 (Flatten)         (None, 256)               0

 dense_2 (Dense)             (None, 50)                12850

 dense_3 (Dense)             (None, 1)                 51

=================================================================
Total params: 1451621 (5.54 MB)
Trainable params: 1451621 (5.54 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 4.17 LSTM 2-Class 256 Units

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 1, 256)            291840

 dropout_4 (Dropout)         (None, 1, 256)            0

 gru_1 (GRU)                 (None, 1, 256)            394752

 dropout_5 (Dropout)         (None, 1, 256)            0

 gru_2 (GRU)                 (None, 1, 256)            394752

 flatten_2 (Flatten)         (None, 256)               0

 dense_4 (Dense)             (None, 50)                12850

 dense_5 (Dense)             (None, 1)                 51

=================================================================
Total params: 1094245 (4.17 MB)
Trainable params: 1094245 (4.17 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 4.18 GRU 2-Class 256 Units

4.5.3 5-Class of Model Summary

In the figure 4.19, 4.20 and 4.21, these are the model summary for RNN, LSTM and GRU for 5-class 64 units parameters.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 1, 64)             11968

 dropout (Dropout)           (None, 1, 64)             0

 simple_rnn_1 (SimpleRNN)    (None, 1, 64)             8256

 dropout_1 (Dropout)         (None, 1, 64)             0

 simple_rnn_2 (SimpleRNN)    (None, 1, 64)             8256

 flatten (Flatten)           (None, 64)                0

 dense (Dense)               (None, 50)                3250

 dense_1 (Dense)             (None, 5)                 255

=================================================================
Total params: 31985 (124.94 KB)
Trainable params: 31985 (124.94 KB)
Non-trainable params: 0 (0.00 Byte)
```


Figure 4.19 RNN 5-Class 64 Units

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 1, 64)             47872

 dropout_2 (Dropout)         (None, 1, 64)             0

 lstm_1 (LSTM)               (None, 1, 64)             33024

 dropout_3 (Dropout)         (None, 1, 64)             0

 lstm_2 (LSTM)               (None, 1, 64)             33024

 flatten_1 (Flatten)         (None, 64)                0

 dense_2 (Dense)             (None, 50)                3250

 dense_3 (Dense)             (None, 5)                 255

=================================================================
Total params: 117425 (458.69 KB)
Trainable params: 117425 (458.69 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 4.20 LSTM 5-Class 64 Units


```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 1, 64)             36096

 dropout_4 (Dropout)         (None, 1, 64)             0

 gru_1 (GRU)                 (None, 1, 64)             24960

 dropout_5 (Dropout)         (None, 1, 64)             0

 gru_2 (GRU)                 (None, 1, 64)             24960

 flatten_2 (Flatten)         (None, 64)                0

 dense_4 (Dense)             (None, 50)                3250

 dense_5 (Dense)             (None, 5)                 255

=================================================================
Total params: 89521 (349.69 KB)
Trainable params: 89521 (349.69 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 4.21 GRU 5-Class 64 Units

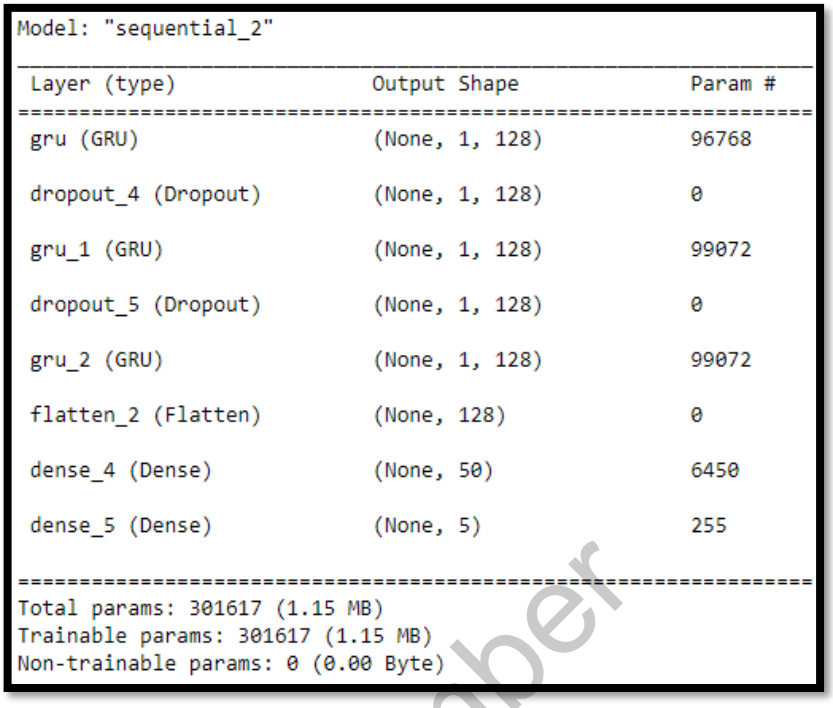
In the figure 4.22, 4.23 and 4.24, these are the model summary for RNN, LSTM and GRU for 5-class 128 units parameters.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 1, 128)            32128

 dropout (Dropout)           (None, 1, 128)            0

 simple_rnn_1 (SimpleRNN)    (None, 1, 128)            32896

 dropout_1 (Dropout)         (None, 1, 128)            0

 simple_rnn_2 (SimpleRNN)    (None, 1, 128)            32896

 flatten (Flatten)           (None, 128)               0

 dense (Dense)               (None, 50)                6450

 dense_1 (Dense)             (None, 5)                 255

=================================================================
Total params: 104625 (408.69 KB)
Trainable params: 104625 (408.69 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 4.22 RNN 5-Class 128 Units

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 1, 128)            128512

 dropout_2 (Dropout)         (None, 1, 128)            0

 lstm_1 (LSTM)               (None, 1, 128)            131584

 dropout_3 (Dropout)         (None, 1, 128)            0

 lstm_2 (LSTM)               (None, 1, 128)            131584

 flatten_1 (Flatten)         (None, 128)               0

 dense_2 (Dense)             (None, 50)                6450

 dense_3 (Dense)             (None, 5)                 255

=================================================================
Total params: 398385 (1.52 MB)
Trainable params: 398385 (1.52 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 4.23 LSTM 5-Class 128 Units



```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 1, 128)            96768

 dropout_4 (Dropout)         (None, 1, 128)            0

 gru_1 (GRU)                 (None, 1, 128)            99072

 dropout_5 (Dropout)         (None, 1, 128)            0

 gru_2 (GRU)                 (None, 1, 128)            99072

 flatten_2 (Flatten)         (None, 128)               0

 dense_4 (Dense)             (None, 50)                6450

 dense_5 (Dense)             (None, 5)                 255

=================================================================
Total params: 301617 (1.15 MB)
Trainable params: 301617 (1.15 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 4.24 GRU 5-Class 128 Units

In the figure 4.25, 4.26 and 4.27, these are the model summary for RNN, LSTM and GRU for 5-class 256 units parameters.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 1, 256)            97024

 dropout (Dropout)           (None, 1, 256)            0

 simple_rnn_1 (SimpleRNN)    (None, 1, 256)            131328

 dropout_1 (Dropout)         (None, 1, 256)            0

 simple_rnn_2 (SimpleRNN)    (None, 1, 256)            131328

 flatten (Flatten)           (None, 256)               0

 dense (Dense)               (None, 50)                12850

 dense_1 (Dense)             (None, 5)                 255

=================================================================
Total params: 372785 (1.42 MB)
Trainable params: 372785 (1.42 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 4.25 RNN 5-Class 256 Units

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 1, 256)            388096

 dropout_2 (Dropout)         (None, 1, 256)            0

 lstm_1 (LSTM)               (None, 1, 256)            525312

 dropout_3 (Dropout)         (None, 1, 256)            0

 lstm_2 (LSTM)               (None, 1, 256)            525312

 flatten_1 (Flatten)         (None, 256)               0

 dense_2 (Dense)             (None, 50)                12850

 dense_3 (Dense)             (None, 5)                 255

=================================================================
Total params: 1451825 (5.54 MB)
Trainable params: 1451825 (5.54 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 4.26 LSTM 5-Class 256 Units

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 1, 256)            291840

 dropout_4 (Dropout)         (None, 1, 256)            0

 gru_1 (GRU)                 (None, 1, 256)            394752

 dropout_5 (Dropout)         (None, 1, 256)            0

 gru_2 (GRU)                 (None, 1, 256)            394752

 flatten_2 (Flatten)         (None, 256)               0

 dense_4 (Dense)             (None, 50)                12850

 dense_5 (Dense)             (None, 5)                 255

=================================================================
Total params: 1094449 (4.17 MB)
Trainable params: 1094449 (4.17 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 4.27 GRU 5-Class 256 Units