KANTIPUR ENGINEERING COLLEGE

(Affiliated to Tribhuvan University)

Dhapakhel, Lalitpur

[Subject Code: CT755]

A MAJOR PROJECT FINAL REPORT ON

NETWORK INTRUSION DETECTION SYSTEM USING LSTM

Submitted by:

Aman Devkota [KAN076BCT010]

Ankur Karmacharya [KAN076BCT013]

Prashad Adhikary [KAN076BCT056]

A MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

Submitted to:

Department of Computer and Electronics Engineering

February, 2024

NETWORK INTRUSION DETECTION SYSTEM USING LSTM

Submitted by:

Aman Devkota [KAN076BCT010]

Ankur Karmacharya [KAN076BCT013]

Prashad Adhikary [KAN076BCT056]

Supervised by:

Dr. Babu Ram Dawadi

Asst. Professor

Department of Electronics and Computer Engineering, IOE Pulchowk

A MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

Submitted to:

Department of Computer and Electronics Engineering

Kantipur Engineering College

Dhapakhel, Lalitpur

February, 2024

ABSTRACT

Network intrusion attacks have significantly increased in recent years, which creates serious privacy and security concerns. Technology development has made cyber-security threats more sophisticated, to the point where the detection mechanisms in place are unable to handle the problem. Therefore, the key to solving this issue would be the deployment of a clever and efficient network intrusion detection system. In order to create an intelligent detection system that can recognize many types of network intrusions, we are using deep learning technique in this project, namely Long Short-Term Memory (LSTM). One of the most recent realistic dataset, InSDN datasets, was used for training and evaluation in order to demonstrate the effectiveness of the suggested system. To evaluate the model, we have used evaluation metrics like accuracy, precision, True Positive Rate, False Positive Rate.

Keywords *– LSTM,Network Intrusion Detection System (NIDS), InSDN dataset*

ACKNOWLEDGMENT

This project is prepeared in partial fulfillment of the requirement for the bachelor's degree in Computer Engineering. We are very grateful to Prof. Dr. Babu Ram Dawadi, our supervisor for guiding us throughout this major project. We would like to express our sincere gratitude to Department head Er. Rabindra Khati, Project Co-ordinator Er. Bishal Thapa and all the faculty members of Kantipur Engineering College for the continuous support during this project for their patience, motivation,enthusiasm, and immense knowledge. Their guidance helped us in all time of research, development and implementation of this project.

Finally we would like to thank our family and friends for all the support and encouragement for the completion of this project.

Aman Devkota	[KAN076BCT010]
Ankur Karmacharya	[KAN076BCT013]
Prashad Adhikary	[KAN076BCT056]

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

DOS: Denial of Service

FN: False Negative

FP: False Positive

IDS: Intrusion Detection System

LSTM: Long Short-Term Memory

RFC: Random Forest Classifier

RFE: Recursive Feature Elimination

R2L: Root to Local

SDN: Software Defined Networking

TN: True Negative

TP: True Positive

U2R: User to Root

CHAPTER 1

INTRODUCTION

1.1 Background

With the increasingly deep integration of the Internet and society, the Internet is changing the way in which people live, study and work, but the various security threats that we face are becoming more and more serious. How to identify various network attacks, especially unforeseen attacks, is an unavoidable key technical issue. An Intrusion Detection System (IDS), a significant research achievement in the information security field, can identify an invasion, which could be an ongoing invasion or an intrusion that has already occurred. In fact, intrusion detection is usually equivalent to a classification problem, such as a binary or a multi class classification problem, i.e., identifying whether network traffic behavior is normal or anomalous, or a five-category classification problem, i.e., identifying whether it is normal or any one of the other four attack types: Denial of Service (DOS), User to Root (U2R), Probe (Probing) and Root to Local (R2L). In short, the main motivation of intrusion detection is to improve the accuracy of classifiers in effectively identifying the intrusive behavior.

Machine learning methodologies have been widely used in identifying various types of attacks, and a machine learning approach can help the network administrator take the corresponding measures for preventing intrusions. However, most of the traditional machine learning methodologies belong to shallow learning and often emphasize feature engineering and selection; they cannot effectively solve the massive intrusion data classification problem that arises in the face of a real network application environment. With the dynamic growth of data sets, multiple classification tasks will lead to decreased accuracy. In addition, shallow learning is unsuited to intelligent analysis and the forecasting requirements of high-dimensional learning with massive data. In contrast, deep learners have the potential to extract better representations from the data to create much better models. As a result, intrusion detection technology has experienced rapid development after falling into a relatively slow period[1].

A well-known method of securing the network is through implementing an Intrusion Detection System (IDS). IDS was originally implemented in 1980. The main aim of their work was to introduce a mechanism which differentiates between benign activities from malicious ones. Further research was carried out to optimizing this methodology to aid monitoring the network traffic in case of attacks, this system is now known as Network Intrusion Detection System (NIDS) . In NIDS, the detection system is inspecting the incoming and outgoing network traffic from all hosts in real time and based on certain criteria, it can detect and identify the attack, then, take the suitable security measures to stop or block it, which significantly reduces the risk of damage to the network. However, due to the rapid increase in the complexity of the cyber-security attacks, the current methods used in NIDS are failing to sufficiently address this issue[2].

IDSs can be divided into two categories according to the main detection technology: misuse detection and anomaly detection. Misuse detection is a knowledge-based detection technology. A misuse detection system needs to clearly define the features of the intrusion, then identify the intrusion by matching the rules. Misuse detection can achieve a high accuracy and low false alarm rate. However, it needs to build a feature library and cannot detect unknown attacks. In contrast, anomaly detection is a behavior-based detection technology. First, it needs to define the normal activities of a network, and then check whether the actual behavior has deviated from the normal activities. Anomaly detection needs only to define a normal state of a specific network, without prior knowledge of intrusion. Thus, it can detect unknown attacks, although there may be a high false alarm rate. At present, network structure is becoming more and more complicated, and intrusion methods are following the trend of diversification and complication, creating more challenges for IDSs.

The recurrent neural network (RNN) has failed to become a mainstream network model in the past few years due to difficulties in training and computational complexity. In recent years, with the development of deep learning theory, RNN began to enter a rapid development period. Currently, RNN has already been applied successfully to handwriting and speech recognition. The main feature of RNN is that it circulates information in a hidden layer which can remember information processed previously, leading to a structural advantage for the processing of time series information. Correspondingly, many intrusion behaviors can be abstracted as specific time series of events from the underlying network. So, RNN is considered suitable for building an IDS[3].

1.2 Problem Statement

The existing Network Intrusion Detection Systems face challenges in accurately and efficiently detecting and preventing network intrusions, thereby compromising the overall security of computer networks. These challenges include high false-positive rates, limited scalability, inability to detect novel or sophisticated attacks, and the difficulty in distinguishing between legitimate and malicious network traffic. Addressing these issues is crucial to enhance the performance and reliability of NIDS, enabling proactive identification and prevention of network intrusions while minimizing false alarms.

1.3 Objectives

The primary objective of this project:

i. To develop a robust and accurate system that can effectively detect and notify network intrusions or malicious activities within a computer network.

1.4 Project Features

The project will be able to accomplish following:

- Anomaly Detection
- Real-time Monitoring
- Traffic Preprocessing

1.5 Application Scope

Network Intrusion Detection System has various applications in areas such as Enterprise networks, Internet Service Providers(ISPs), Cloud Computing environment, Government networks and so on. The application scope of network intrusion detection systems using deep neural networks extends beyond these areas, as network security is crucial in nearly every sector that relies on secure and reliable communication. By deploying these systems, organizations can proactively detect and respond to network intrusions, minimize the impact of attacks, and protect their assets, data, and operations from potential threats.

1.6 System Requirement

1.6.1 Development Requirements

1.6.1.1 Software Requirements

- Windows/Linux/Mac
- VMware/Virtualbox
- Mininet, POX controller
- Jupyter Notebook
- Python IDE

1.6.1.2 Hardware Requirements

- PC with at least 4-8 GB RAM
- Higher graphics of at least 2 GB

1.6.2 Deployment Requirements

1.6.2.1 Software Requirements

- Windows/Linux/Mac
- VMware/Virtualbox
- Mininet, POX controller

1.6.2.2 Hardware Requirements

- More than 1.5 GHz clock speed
- Minimum 4 GB RAM

1.7 Project Feasibility

1.7.1 Technical Feasibility

The technical feasibility assessment is focused on gaining in understanding of the present technical resources required by the system and their applicability to the expected needs of the proposed system. Regarding the proposed system, the technical requirement includes a PC.

1.7.2 Operational Feasibility

The user will not need any formal knowledge about programming so our project is operationally feasible.

1.7.3 Economic Feasibility

The purpose of the economic feasibility assessment is to determine the positive economic benefits to the user that the proposed system will provide. Most of the software used for the development is free. Thus, the project is economically feasible.

1.7.4 Schedule Feasibility

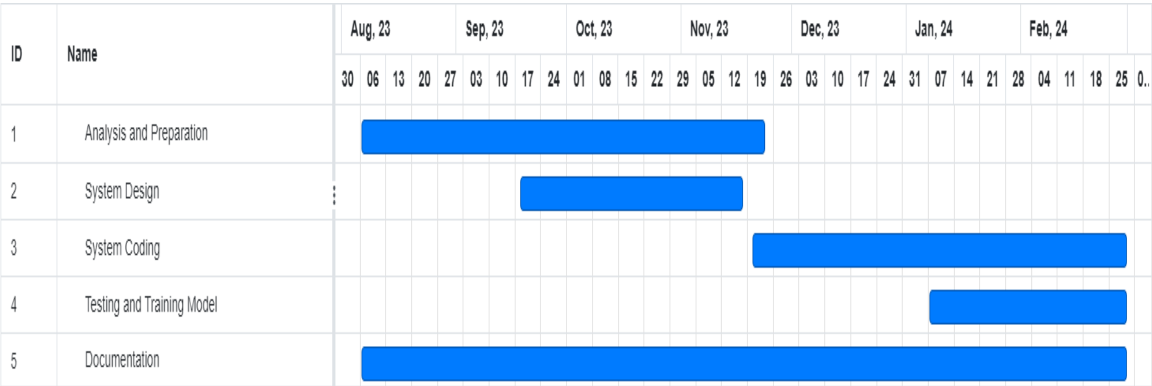

Figure 1.1: Gantt Chart

CHAPTER 2
LITERATURE REVIEW
2.1 Related Projects
2.1.1 Suricata

Suricata is a leading open-source network analysis and threat detection tool, widely recognized for its high performance and versatility across various platforms including Windows, Mac, Unix, and Linux. Developed by the Open Information Security Foundation (OSIF), Suricata is embraced by a broad spectrum of users from private and public sectors to major vendors seeking to bolster their cybersecurity defenses. Its utility spans enterprises of all sizes, offering a cost-effective solution for intricate security monitoring needs.

Distinguished by its dual functionality, Suricata serves not only as an intrusion detection system (IDS) that identifies and alerts on potential threats but also as an intrusion prevention system (IPS) with capabilities to actively intervene and mitigate identified threats by blocking malicious traffic. This dual capability sets Suricata apart in the cybersecurity domain, enabling comprehensive protection against a wide array of cyber threats.

One of the core strengths of Suricata lies in its sophisticated rule set and signature language, which allows for precise threat detection and prevention strategies. Additionally, its proficiency in deep packet inspection (DPI) further enhances its effectiveness, making it an ideal solution for a variety of security monitoring initiatives. Whether it's for detecting subtle anomalies or preventing advanced cyber attacks, Suricata's robust feature set ensures that organizations can safeguard their digital assets effectively. As a result, Suricata stands out as a versatile and powerful tool in the arsenal of cybersecurity professionals, offering a free yet invaluable resource for securing network environments against the ever-evolving landscape of cyber threats.

2.1.2 Snort

Snort stands as the premier Open Source Intrusion Prevention System (IPS) globally, renowned for its versatility and comprehensive security capabilities. At its core, Snort operates by implementing a robust set of rules designed to identify and define malicious network activities. These rules are crucial for detecting packets that exhibit patterns or behaviors indicative of cyber threats, triggering alerts for users to take action. Moreover, Snort's functionality extends beyond mere detection; it can be configured to operate inline, actively intercepting and blocking malicious packets before they infiltrate the network.

Snort's flexibility allows it to serve multiple roles within the cybersecurity framework. It can function as a packet sniffer, akin to tcpdump, offering detailed insights into network traffic for analysis and troubleshooting purposes. Additionally, it serves as an effective packet logger, providing a valuable tool for network traffic debugging by recording data packets traversing the network. Most prominently, Snort excels as a full-fledged network intrusion prevention system, offering robust protection against a wide array of cyber threats.

Available for both personal and business use, Snort can be easily downloaded and configured to suit various security needs. Its open-source nature not only facilitates widespread access but also encourages continuous improvement and updates from a global community of developers and cybersecurity experts. This collaborative approach ensures that Snort remains at the forefront of intrusion prevention technology, providing an essential layer of defense for networks around the world against the ever-evolving landscape of cyber threats.

2.2 Related Works

The paper titled "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks" introduces a cutting-edge approach to enhancing the efficacy of Intrusion Detection Systems (IDS) through the implementation of Recurrent Neural Networks (RNN-IDS). This innovative methodology diverges from traditional machine learning classification techniques by leveraging the dynamic and temporal processing capabilities inherent in RNNs, which are particularly adept at handling sequence data, thus making them an ideal choice for analyzing network traffic and detecting anomalous patterns indicative of cyber threats.

The core of this research revolves around the comparative analysis between RNN-IDS and conventional machine learning methods across both binary and multiclass classification scenarios. The comparison aimed to rigorously evaluate the performance of the deep learning-based model in terms of its accuracy in detecting intrusions within network systems. Notably, the findings from this study illuminated the superior performance of the RNN-IDS model, demonstrating a significant improvement in detection accuracy over its traditional counterparts.

This improvement is attributed to the RNN-IDS model's ability to capture and learn from the temporal dependencies present in network traffic data, a critical aspect that traditional machine learning models often overlook. By effectively utilizing the sequential information within the data, RNN-IDS can offer a more nuanced and comprehensive analysis of potential security threats, leading to a higher accuracy rate in intrusion detection.

Moreover, the success of the RNN-IDS model in outperforming established machine learning techniques not only validates the potential of deep learning approaches in the realm of cyber-security but also paves the way for further research into the development of more sophisticated and effective intrusion detection systems. The study concludes by highlighting the RNN-IDS model as a novel and promising research method, capable of significantly enhancing the accuracy and reliability of intrusion detection efforts, thereby contributing to the advancement of security measures in the face of evolving cyber threats[1].

The study "Using Deep Learning Techniques for Network Intrusion Detection" delves into the development of an advanced intrusion detection system leveraging the strengths of both Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). This innovative approach aimed to harness the unique capabilities of these deep learning models to enhance the detection of various network intrusions, a critical aspect of maintaining cybersecurity in the digital age. The evaluation of this intelligent detection system was conducted through a comprehensive analysis utilizing a range of performance metrics, including accuracy, F1 score, recall, and precision. These metrics served as the benchmarks to assess and compare the effectiveness of the employed learning techniques in identifying and classifying malicious activities within network traffic.

The comparative analysis revealed that CNNs emerged as the superior model, demonstrating exceptional performance across all evaluated metrics. This outcome underscores the effectiveness of CNNs in feature extraction and pattern recognition within complex datasets, attributes that are essential for the accurate detection of network intrusions. The CNN model's ability to outperform other techniques signifies its robustness and reliability as a tool for enhancing the security infrastructure against diverse cyber threats.

This finding not only highlights the potential of deep learning in revolutionizing network intrusion detection but also sets a precedent for future research in cybersecurity. By identifying CNN as the most effective model for intrusion detection, the study contributes valuable insights into the optimization of security systems, paving the way for the development of more resilient and intelligent cybersecurity solutions[2].

The research paper titled "An Intrusion Detection System Using a Deep Neural Network With Gated Recurrent Units" introduces a groundbreaking approach to enhancing intrusion detection systems (IDS) through the application of deep learning theories, specifically focusing on the capabilities of Gated Recurrent Units (GRU). This innovative IDS model incorporates automatic feature extraction, utilizing a sophisticated combination of GRU, Multilayer Perceptron (MLP), and a softmax module to effectively analyze and detect time-related network intrusions. The proposed system's architecture leverages the sequential data processing strength of GRUs, a variant of recurrent neural networks, to capture dynamic changes in network traffic that may indicate malicious activities.

Through rigorous experimentation with relevant datasets, the study demonstrates the superior performance of the GRU-based IDS model over traditional methods. The research further compares GRU with Long Short-Term Memory (LSTM) units, another popular recurrent neural network variant. The findings clearly indicate that GRUs offer a more suitable memory mechanism for IDS applications than LSTMs, attributing this to GRUs' simplified yet highly efficient structure, which enhances its capability for pattern recognition in network security.

Significantly, the paper highlights the exceptional efficacy of bidirectional GRU configurations, which outperform other tested models. This suggests that the bidirectional approach maximizes the model's ability to understand and predict intrusion scenarios by analyzing data from both past and future contexts, providing a more comprehensive detection mechanism. The success of the GRU-based model marks a significant advancement in the field of cybersecurity, offering a potent tool for the development of more accurate, efficient, and reliable intrusion detection systems[3].

The paper "A Flow-Based Anomaly Detection Approach With Feature Selection Method Against DDoS Attacks in SDNs" embarks on a comprehensive exploration of Distributed Denial of Service (DDoS) attack detection methodologies within Software-Defined Networking (SDN) environments. This scholarly work seeks to thoroughly examine the landscape of existing research in the realm of DDoS defense mechanisms, specifically tailored to the unique architectural nuances of SDNs. The primary objective is to unearth the existing knowledge gaps and underscore the imperative for continued investigative efforts in this critical area of cybersecurity.

Central to this study is the establishment of a robust theoretical framework that underpins the proposed anomaly detection approach. By meticulously selecting relevant features that are indicative of DDoS attack patterns, the research endeavors to craft a sophisticated detection model that stands on the shoulders of its predecessors while pioneering new frontiers in attack mitigation strategies. This involves a detailed comparative analysis with prior models to not only validate the proposed method's efficacy but also to illuminate its contributions to advancing the state of the art in DDoS detection.

Furthermore, the document provides an insightful context that encapsulates the significance of the research undertakings, delineating the practical and theoretical implications of the findings. By articulating the relevance of the study, it aims to foster a deeper understanding among the academic and professional communities about the criticality of securing SDN infrastructures against the burgeoning threat of DDoS attacks. This comprehensive approach not only reinforces the need for innovative research in this domain but also sets the stage for future advancements that could revolutionize the way DDoS attacks are detected and mitigated in increasingly dynamic network environments[4].

The innovative research detailed in the paper ”Securing IoT and SDN Systems Using Deep-Learning Based Automatic Intrusion Detection” presents the Secured Automatic Two-level Intrusion Detection System (SATIDS), a cutting-edge solution leveraging the advanced capabilities of Long Short-Term Memory (LSTM) networks for bolstering cybersecurity measures in Internet of Things (IoT) and Software-Defined Networking (SDN) infrastructures. This system represents a significant leap forward in the use of deep learning techniques for cyber defense, specifically employing LSTM networks known for their proficiency in handling sequential data and their ability to uncover complex patterns indicative of cyber threats.

SATIDS distinguishes itself by introducing an enhanced LSTM model designed to meticulously analyze network traffic, segregating it into normal behavior and potential threats. This differentiation is pivotal, as it not only flags immediate security concerns but also categorizes the nature of the attack and pinpoints its specific subtype, enabling a more targeted response to the intrusion. The deployment of SATIDS involves rigorous training and validation phases, utilizing comprehensive datasets like ToN-IoT and InSDN, which are reflective of real-world network environments and attack scenarios. This methodological approach ensures the system's effectiveness and reliability in detecting a wide array of cyber threats.

Upon comparison with existing Intrusion Detection Systems (IDS), SATIDS has demonstrated superior performance across several critical metrics, including accuracy, precision, F1-score, and overall detection rate. Such results highlight the system's exceptional capability in differentiating between benign and malicious traffic flows accurately, an essential feature for ensuring network security. Notably, SATIDS has proven highly effective in identifying and responding to sophisticated attack vectors, such as backdoor and Distributed Denial of Service (DDoS) attacks, that pose significant risks to IoT and SDN environments.

As a whole, the SATIDS framework, underpinned by an advanced LSTM network, offers a comprehensive and effective strategy for intrusion detection within IoT and SDN settings. Its ability to accurately detect and classify network anomalies represents a substantial advancement in cybersecurity technology, providing enhanced protection for complex networked systems against an evolving landscape of cyber threats. This research not only underscores the potential of deep learning in cybersecurity but also sets a new benchmark for the development of intelligent, adaptive IDS solutions capable of safeguarding the next generation of digital infrastructures[5].

CHAPTER 3
THEORETICAL FRAMEWORK
3.1 LSTM Operation

LSTM is a popular deep learning technique in RNN for time series prediction. While standard RNNs outperform traditional networks in preserving information, they are not very effective in learning long term dependencies due to the vanishing gradient problem. An LSTM is well-suited to classify and/or predict time-series data. There are several architectures of LSTM units. A common architecture is composed of a memory cell, an input gate, an output gate and a forget gate. The mathematical formulation of the LSTM cell is given below:

$$f_t = \sigma(x_t W_f + H_{t-1} U_f) \tag{3.1}$$

$$o_t = \sigma(x_t W_o + H_{t-1} U_o) \tag{3.2}$$

$$S_t = \sigma(S_{t-1} * f_t + i_t * H_t^{'}) \tag{3.3}$$

$$i_t = \sigma(x_t W_i + H_{t-1} U_i) \tag{3.4}$$

$$H_t^{'} = tanh(x_t W_g + H_{t-1} U_g) \tag{3.5}$$

$$H_t = tanh(S_t) * o_t \tag{3.6}$$

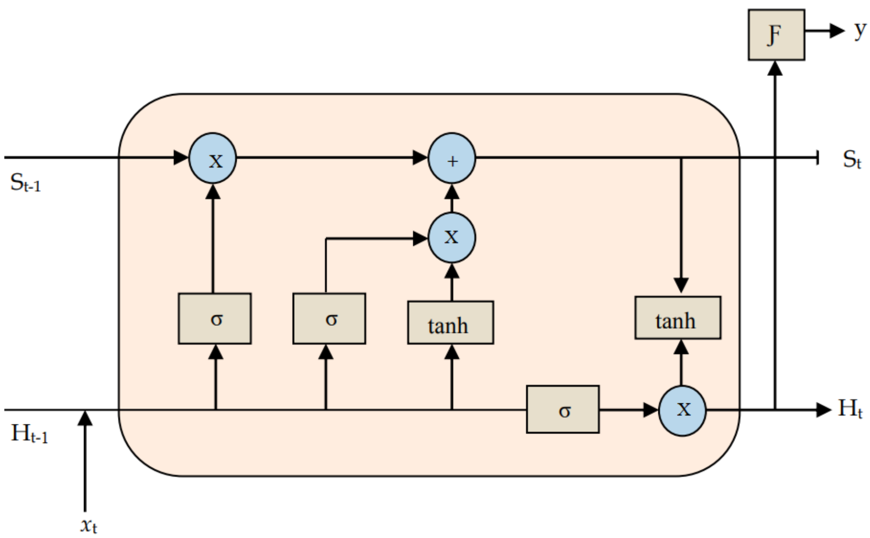

Figure 3.1: LSTM

3.2 Activation Function

The activation function in a neural network is a mathematical function that is applied to a network node's output and decides whether or not the output of the node should be activated based on the weighted total of the inputs. Softmax activation function was employed in the project's models.

3.2.1 Softmax

Softmax is an activation function that scales numbers/logits into probabilities. The output of a Softmax is a vector (say v) with probabilities of each possible outcome. The probabilities in vector v sums to one for all possible outcomes or classes. Mathematically, Softmax is defined as:

$$S(z_i) = \frac{exp(z_i)}{\sum_{j=1}^{n} exp(z_j)} \tag{3.7}$$

where z is an input vector to a softmax function,S, n is number of classes(possible outcomes), denominator denotes a normalization term whereas the numerator denotes the standard exponential function applied on the i-th element of the input vector.

3.3 Categorical Cross-Entropy

Categorical cross-entropy loss is utilized in multi-class classification tasks with more than two mutually exclusive classes. Similarly to the binary, this type of cross-entropy loss function quantifies the dissimilarity between the predicted probabilities and the true categorical labels. The categorical cross-entropy loss function is commonly used in neural networks with softmax activation in the output layer for multi-class classification tasks. By minimizing loss, the model learns to assign higher probabilities to the correct class while reducing the probabilities for incorrect classes, improving accuracy.

3.4 Adam Optimizer

The Adam optimizer is a popular optimization algorithm used in training deep neural networks. It combines the advantages of two other optimization techniques: AdaGrad and RMSProp. Adam maintains adaptive learning rates for each parameter by calculating an exponentially decaying average of past gradients and their squares. This allows it to dynamically adjust the learning rate for each parameter, typically resulting in faster convergence and better performance compared to traditional optimization algorithms. Adam also includes bias correction to prevent the initial steps from being too large. Overall, Adam is well-suited for a wide range of deep learning tasks due to its efficiency, robustness, and ease of use, often requiring minimal hyper parameter tuning.

3.5 Dropout

Dropout is a regularization technique for neural networks, introduced by Geoffrey Hinton, et.al, in 2012. Dropout involves randomly setting a fraction of activation of neurons to 0. This reduces the amount of information available to each layer, forcing the network to learn multiple independent representations of the same data. This makes the network more robust to overfitting. In practice, during each forward pass, each activation in the network is set to zero with a certain probability (e.g., 50%), effectively dropping out that activation and its corresponding nodes in the network. During the backward pass, the gradients are computed normally and then multiplied by a factor that corresponds to the keep probability. This allows the network to learn to 'turn on' different nodes and combinations of nodes to model the data. In a deep neural network architecture, dropout layers are inserted between the dense layers or the convolution layers. The keep probability is typically set to a value between 0.5 and 0.8, depending on the size and complexity of the network and the size of the training data.

3.6 Evaluation Metrics

The most important performance indicator (Accuracy, AC) of intrusion detection is used to measure the performance of the model. In addition to the accuracy, we introduce the detection rate and false positive rate. The True Positive (TP) is equivalent to those correctly rejected, and it denotes the number of anomaly records that are identified as anomaly. The False Positive (FP) is the equivalent of incorrectly rejected, and it denotes the number of normal records that are identified as anomaly. The True Negative (TN) is equivalent to those correctly admitted, and it denotes the number of normal records that are identified as normal. The False Negative (FN) is equivalent to those incorrectly admitted, and it denotes the number of anomaly records that are identified as normal.

Predicted

	Classes	DDoS	Normal	Probe
Actual	DDoS	TN	FP	TN
	Normal	FN	TP	FN
	Probe	TN	FP	TN

Figure 3.2: Confusion Matrix

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.8}$$

Precision is the number of actual attacks as a proportion of the number classified as attacks.

$$Precision = \frac{TP}{TP + FP} \tag{3.9}$$

True Positive Rate or Recall shows the percentage of the number of records identified correctly over the total number of anomaly records.

$$TruePositiveRate/Recall = \frac{TP}{FN + TP} \tag{3.10}$$

False Positive Rate is the percentage of the number of records rejected incorrectly is divided by the total number of normal records.

$$FalsePositiveRate = \frac{FP}{FP + TN} \tag{3.11}$$

The F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account.

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3.12}$$

3.7 Software Defined Networking

Software-Defined Networking (SDN) is a revolutionary network architecture approach that decouples the control plane from the data plane in traditional networking models. By centralizing control functions in a software-based controller, SDN enables dynamic management and programmability of network resources. This separation allows administrators to orchestrate network behaviour centrally, facilitating rapid provisioning, configuration, and optimization of network services. SDN simplifies network management by abstracting underlying infrastructure complexities and providing a unified interface for network automation and orchestration. Through programmable interfaces and open standards like OpenFlow, SDN empowers organizations to tailor network behaviour to specific application requirements, enhance agility, and support innovative network services.

3.8 Mininet

Mininet is an open-source software emulation platform for creating virtual networks. It allows users to create a realistic network topology on a single machine for testing, research, and educational purposes. Mininet provides a lightweight and easy-to-use environment for experimenting with SDN (Software-Defined Networking) concepts without the need for physical networking hardware. Mininet allows users to define custom network topologies using a simple Python-based API. Users can specify the number and types of nodes (switches, routers, hosts), the links between them, and even customize parameters such as bandwidth, delay, and loss rates. Mininet utilizes Linux container (LXC) technology to create lightweight virtual network devices. Each network element (node) in a Mininet topology runs as a separate process inside a Linux container, providing isolation and efficient resource utilization. Mininet supports integration with various SDN controllers, including OpenDaylight, ONOS, Ryu, POX, and others. Users can choose their preferred SDN controller and configure Mininet to connect to it, enabling experimentation with different controller functionalities and SDN applications.

3.9 POX Controller

POX Controller is an open-source software-defined networking (SDN) controller framework developed in Python. As a controller, it serves as the centralized brain of an SDN network, managing and orchestrating the flow of data traffic within the network. POX Controller enables network administrators to dynamically control and configure network devices, such as switches and routers, by providing a programmable interface for implementing network policies and forwarding rules. Built on the event-driven architecture of Python, POX Controller offers flexibility and extensibility, allowing users to develop custom network applications and services tailored to their specific requirements. It supports various SDN protocols, including OpenFlow, enabling seamless integration with SDN-enabled devices.

3.10 Wireshark

Wireshark is a powerful open-source network protocol analyser that allows users to capture, analyse, and interpret network traffic in real-time or from previously captured data. With its intuitive user interface and extensive protocol support, Wireshark enables users to inspect packets at a granular level, dissecting each one to reveal details such as source and destination addresses, protocols used, packet contents, and timing information. This level of visibility makes Wireshark invaluable for network troubleshooting, as it helps identify and diagnose issues such as connectivity problems, performance bottlenecks, and security threats. Additionally, Wireshark provides advanced features such as customizable display filters, statistical analysis tools, and packet reconstruction capabilities, empowering users to efficiently analyse network behaviour, detect anomalies, and optimize network performance.

3.11 CICFlowMeter

CICflowmeter or the Canadian Institute for Cybersecurity Flow Meter, is a network traffic analysis tool designed to monitor and analyse network flows, providing insights into network behaviour and traffic patterns. Specifically tailored for cybersecurity purposes, CICflowmeter focuses on detecting and analysing network flows related to cyber threats, such as malicious activities, attacks, and anomalies. By capturing and dissecting flow data, including source and destination IP addresses, ports, protocols, and packet sizes, CICflowmeter offers detailed visibility into network traffic, allowing security analysts to identify potential security breaches, intrusions, or suspicious behaviour.

3.12 Probe

A probe attack is a reconnaissance technique used by attackers to gather information about a target system, network, or infrastructure. The term ”probe” refers to the act of probing or investigating the target to identify vulnerabilities, weaknesses, or potential points of entry for further exploitation. The primary objective of a probe attack is to gather information about the target environment without directly causing any damage or disruption. Attackers aim to identify potential security weaknesses that can be exploited in subsequent stages of an attack. Probe attacks typically involve reconnaissance activities where the attacker gathers information about the target. This information can include network topology, system configurations, installed software and services, open ports, and potential entry points into the network.

3.13 DDoS

A Distributed Denial of Service (DDoS) attack is a malicious attempt to disrupt the normal traffic of a targeted server, service, or network by overwhelming it with a flood of internet traffic. Unlike traditional Denial of Service (DoS) attacks, where a single source is used to flood the target, DDoS attacks involve multiple sources, often tens of thousands or more, distributed across the internet. The primary goal of a DDoS attack is to render a targeted system, network, or service unavailable to its users by overwhelming it with a massive volume of traffic. This disrupts the normal operation of the target, causing it to become slow, unresponsive, or completely unavailable. DDoS attacks are executed by a network of compromised devices, often referred to as a botnet. These devices can include computers, servers, Internet of Things (IoT) devices, routers, and even smartphones infected with malware. The attacker gains control of these devices by infecting them with malicious software, such as viruses, worms, or Trojans, turning them into ”bots” or ”zombies” that can be remotely controlled. Once the botnet is established, the attacker instructs the compromised devices to flood the target with a massive volume of traffic. This flood of traffic overwhelms the target’s resources, making it unable to respond to legitimate user requests.

CHAPTER 4
METHODOLOGY

4.1 Working Mechanism

The development of Network Intrusion Detection System involves major steps which is depicted in the diagram given below:

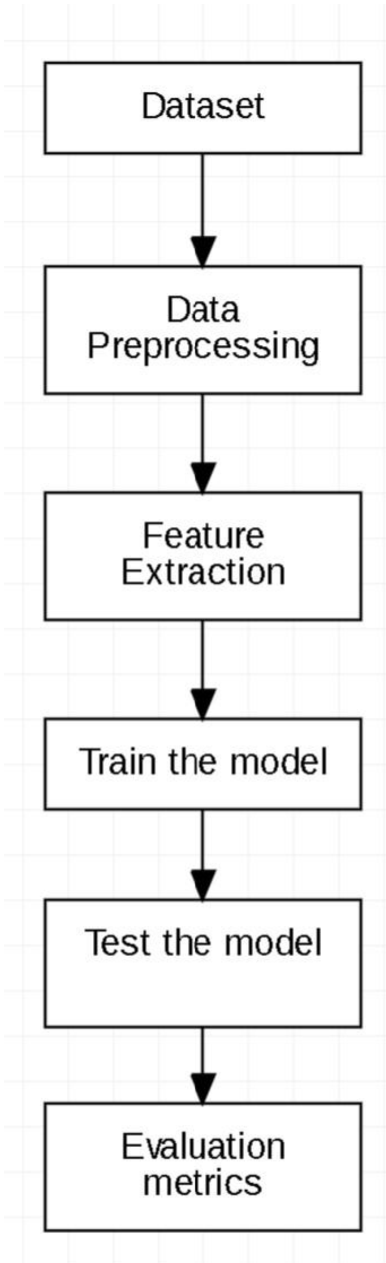

Figure 4.1: Working mechanism of Network Intrusion Detection System

4.1.1 Data set

InSDN is a comprehensive Software-Defined Network (SDN) dataset for Intrusion detection system evaluation. The new dataset includes the benign and various attack categories that can occur in different elements of the SDN standard. InSDN considers different attack, including DoS, DDoS, brute force attack, web applications, exploitation, probe, and botnet. Furthermore, the normal traffic in the generated data covers various popular application services such as HTTPS, HTTP, SSL, DNS, Email, FTP, SSH, etc. The dataset was generated by using four virtual machines (VMs). The first virtual machine is a Kali Linux one and represents the attacker server. The secondary machine is a Ubuntu 16.4 one, and acts on the ONOS controller. Third is an Ubuntu 16.4 machine to serve for the Mininet and OVS switch. The forth virtual machine is a Linux one based on metasploitable-2 to provide vulnerable services for demonstrating common vulnerabilities.[6]

SN	Features	SN	Features
1	Src Port	41	Pkt Len Min
2	Dst Port	42	Pkt Len Max
3	Protocol	43	Pkt Len Mean
4	Flow Duration	44	Pkt Len Std
5	Tot Fwd Pkts	45	Pkt Len Var
6	Tot Bwd Pkts	46	FIN Flag Cnt
7	TotLen Fwd Pkts	47	SYN Flag Cnt
8	TotLen Bwd Pkts	48	RST Flag Cnt
9	Fwd Pkt Len Max	49	PSH Flag Cnt
10	Fwd Pkt Len Min	50	ACK Flag Cnt
11	Fwd Pkt Len Mean	51	URG Flag Cnt
12	Fwd Pkt Len Std	52	CWE Flag Count
13	Bwd Pkt Len Max	53	ECE Flag Cnt
14	Bwd Pkt Len Min	54	Down/Up Ratio
15	Bwd Pkt Len Mean	55	Pkt Size Avg
16	Bwd Pkt Len Std	56	Fwd Seg Size Avg
17	Flow Byts/s	57	Bwd Seg Size Avg
18	Flow Pkts/s	58	Fwd Byts/b Avg
19	Flow IAT Mean	59	Fwd Pkts/b Avg
20	Flow IAT Std	60	Fwd Blk Rate Avg
21	Flow IAT Max	61	Bwd Byts/b Avg
22	Flow IAT Min	62	Bwd Pkts/b Avg
23	Fwd IAT Tot	63	Bwd Blk Rate Avg
24	Fwd IAT Mean	64	Subflow Fwd Pkts
25	Fwd IAT Std	65	Subflow Fwd Byts
26	Fwd IAT Max	66	Subflow Bwd Pkts
27	Fwd IAT Min	67	Subflow Bwd Byts
28	Bwd IAT Tot	68	Init Fwd Win Byts
29	Bwd IAT Mean	69	Init Bwd Win Byts
30	Bwd IAT Std	70	Fwd Act Data Pkts
31	Bwd IAT Max	71	Fwd Seg Size Min
32	Bwd IAT Min	72	Active Mean
33	Fwd PSH Flags	73	Active Std
34	Bwd PSH Flags	74	Active Max
35	Fwd URG Flags	75	Active Min
36	Bwd URG Flags	76	Idle Mean
37	Fwd Header Len	77	Idle Std
38	Bwd Header Len	78	Idle Max
39	Fwd Pkts/s	79	Idle Min
40	Bwd Pkts/s	80	Label

Figure 4.2: Features of InSDN dataset

4.1.2 Data Preprocessing

We used the InSDN datasets and first dropped some label values in the Label column. The label values that were dropped from the dataset were U2R, BFA and DoS. Then we dropped columns like Timestamp, Flow_ID, Src_IP and Dst_IP before the normalization step. For the efficient training of neural networks, the numeric input data should be transformed by performing some pre-processing known as data normalization. It is used where inputs are widely divergent. Without such a process, networks would take a long time to train. Different schemes can be used to normalize the input data before it is fed to the input layer of neural network. We used Z-score normalization to normalize the attributes of our dataset. Z-score normalization refers to the process of normalizing every value in a dataset such that the mean of all of the values is 0 and the standard deviation is 1. Mathematically,

$$Newvalue = \frac{x - \mu}{\lambda} \tag{4.1}$$

where x : Original value,

μ : Mean of data,

λ : Standard deviation of data.

4.1.3 Feature Extraction

Random Forest Classifier, or RFC, is an ensemble learning method that creates a forest of decision trees, where each tree is trained on a random subset of the data, and a random subset of the features. It provides a feature importance score based on how much each feature reduces impurity across all decision trees in the forest. Recursive Feature Elimination, or RFE is a feature selection technique that recursively fits a model and removes the least important features until the desired number of features is reached. It works by repeatedly training the model, ranking the features based on their importance, and removing the least important features. It is particularly useful when the number of features is large, as it helps to reduce the complexity of the model and improves its interpretability. Algorithm for RFC with RFE is given as:

i. Initialise the Random Forest Classifier with the required number of trees.

ii. Use RFE for feature selection taking parameters like RFC as the base estimator and number of features to select.
iii. Fit RFE to the training datasets.
iv. Get the boolean masks of the selected features from RFE.
v. Extract the names of the selected features.
vi. Create a new data frame containing the selected features.

4.1.4 Train and test the model

The model used in this project for NIDS is: Long Short-Term Memory (LSTM).

4.1.4.1 Long Short-Term Memory (LSTM)

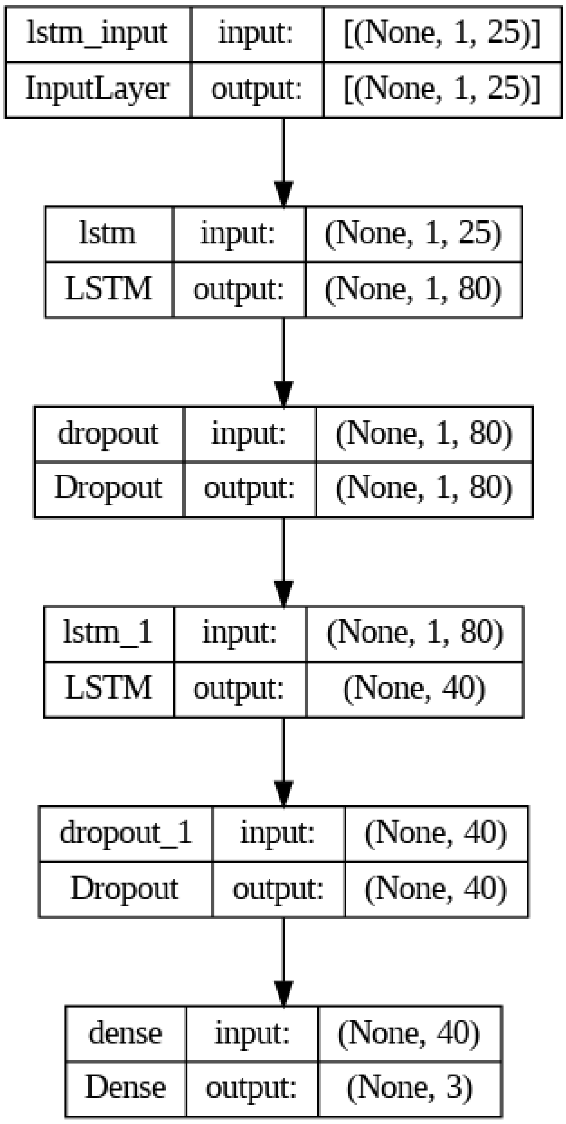

Figure 4.3: LSTM model

The block diagram shows the layers used in the trained LSTM model.

i. LSTM layer: This is the core layer of the model. It is responsible for learning long-term dependencies in the input data. It consists of cells that store information and gates that control the flow of information into and out of the cells.
ii. Dropout layer: This layer is a regularization technique used to prevent the model from overfitting on the training data.
iii. Dense layer: This is the layer responsible for mapping the output of the LSTM layers to a final output.

4.2 Model Training and Optimization

4.2.1 Model Training Algorithm

i. Import the required libraries for data preprocessing including numpy, pandas and scikit-learn.
ii. Using the train-test split function, divide the datasets into training and testing datasets in an 8:2 ratio
iii. Definition of early stopping to prevent over fitting that determines the validation loss defined by a parameter patience.
iv. Compile the model using Adam optimizer, loss function of categorical crossentropy and use accuracy as the metric.
v. Fit the model on the training set with early stopping as callback using the fit() function of the model. Save the object returned by the function in a history parameter.
vi. Plot for training and validation loss against number of epochs.

4.3 System Diagram

4.3.1 Use case diagram

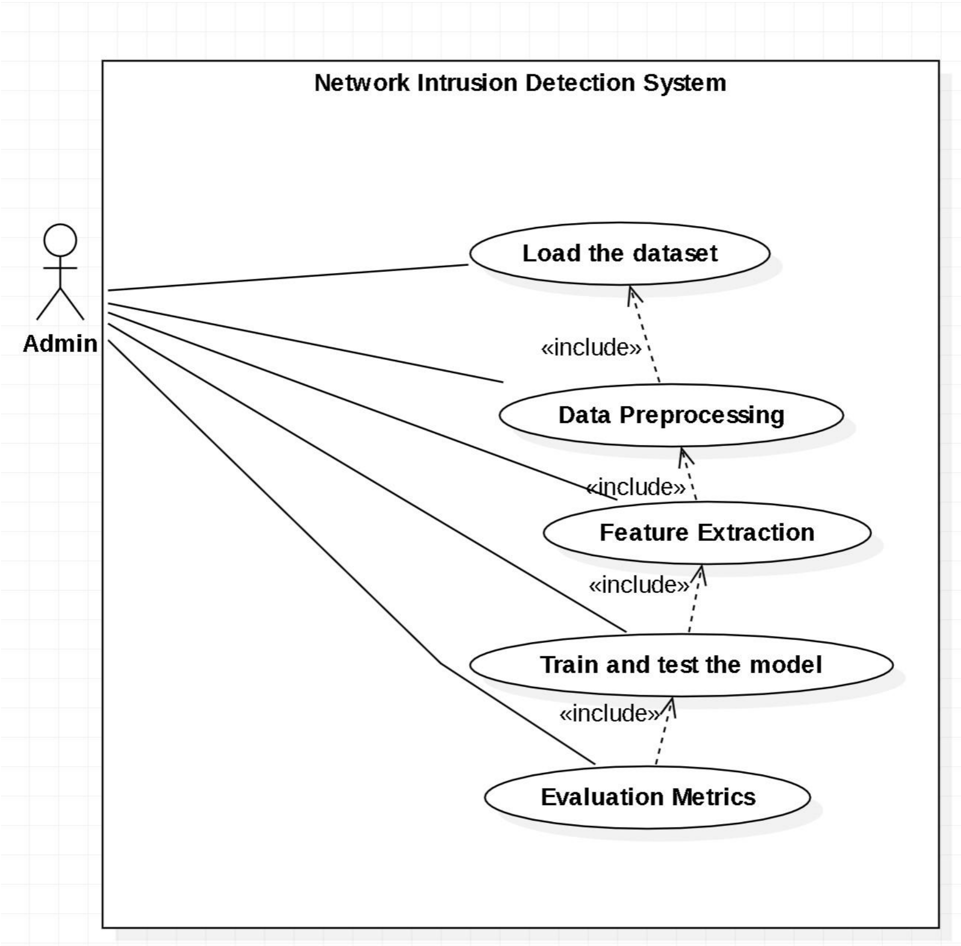

Figure 4.4: Use case Diagram of Network Intrusion Detection System

4.3.2 Simulation Environment

The simulation was created in a virtual machine (Orcale VM VirtualBox) using Ubuntu-22.04.3 as Mininet only support Linux OS. The architecture of our network composes of two virtual switch namely s1 and s2, pox controller c0, Mininet virtual host namely h1, h2, h3, h4, h5 and h6. The virtual switch s1 is connected to Mininet virtual host h1, h2 and h3. The virtual switch s2 is connected to Mininet virtual host h4, h5 and h6. The communication between all virtual hosts is done by L2 connectivity. The legitimate network traffic was generated through Linux machine (Ubuntu-22.04.3) which is connected to the Internet. The DDoS Traffic was generated through Mininet virtual host using hping3 tool. The Probe Traffic was generated through Mininet virtual host using Nmap tool.

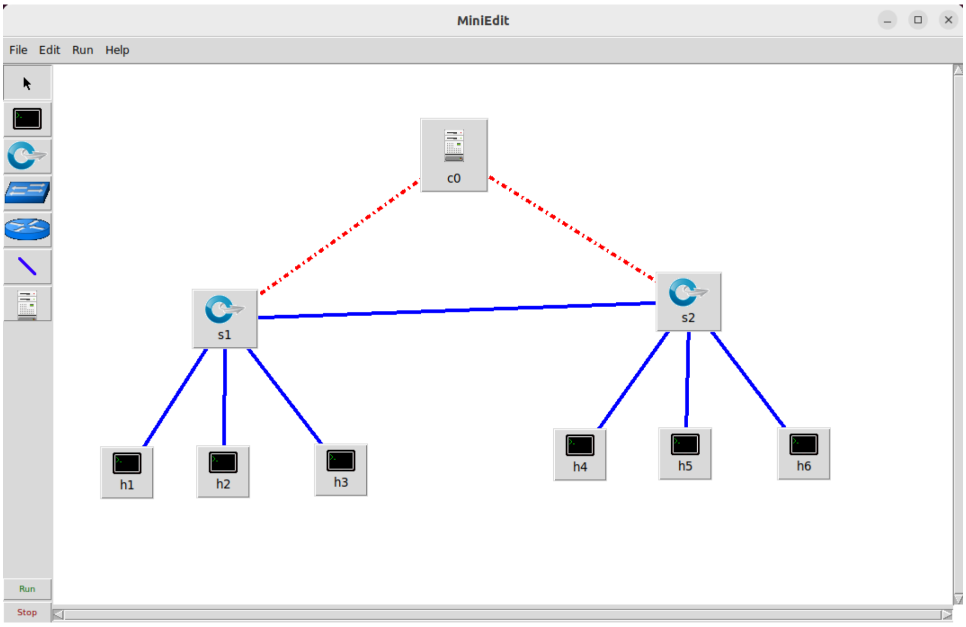

Figure 4.5: Simulation Environment

4.3.3 Software Development Model

Incremental model is a method of software engineering that combines the elements of waterfall model in iterative manner. It involves both development and maintenance. In this model requirements are broken down into multiple modules. Incremental development is done in steps from analysis design, implementation, testing/verification, maintenance. Each iteration passes through the requirements, design, coding and testing phases. The first increment is often a core product where the necessary requirements are addressed, and the extra features are added in the next increments. The core product is delivered to the client. Once the core product is analyzed by the client, there is plan development for the next increment.

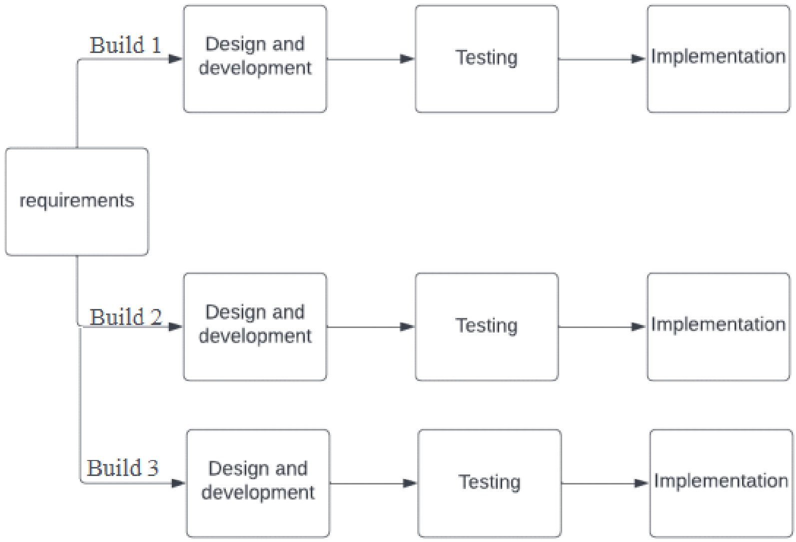

Figure 4.6: Incremental Model

- First build: The feature extraction process was done where features were reduced from 81 to 25 and demo model was built.
- Second build: The model was optimized and validated through test data.
- Third build: The model was tested with real world traffic and integrated with SDN.

CHAPTER 5
RESULTS AND DISCUSSION

We have completed the development of the project along with obtaining the desirable output. We divided the original dataset into train and test set with 80/20 split. We then trained the LSTM model for the train set and checked the classification report and confusion matrix for both the test set and the validation set that we created.

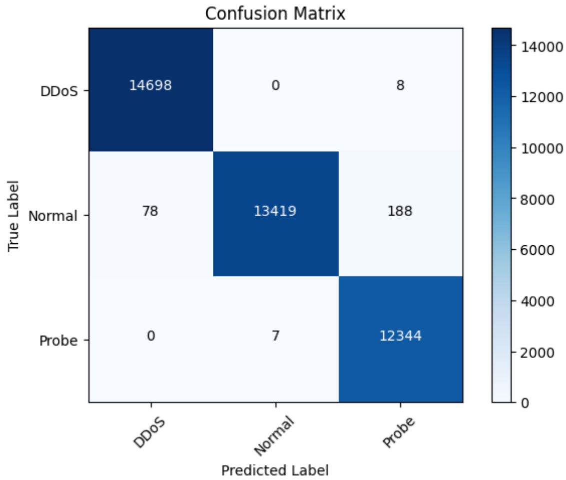

Figure 5.1: Confusion Matrix for Test dataset

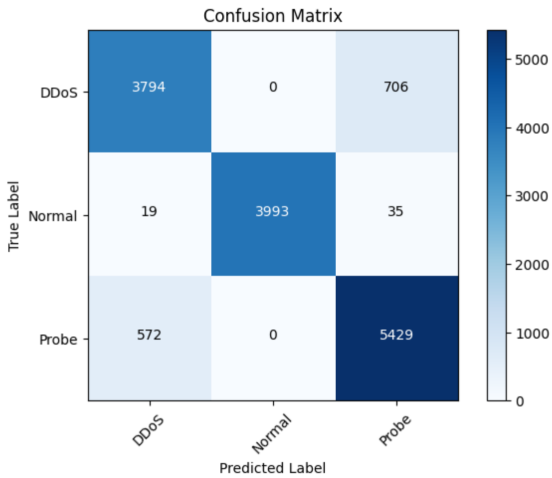

Figure 5.2: Confusion Matrix for Validation dataset

In the above figures, 0 denotes DDoS class, 1 denotes Normal class and Probe class is denoted by 2. As we can see, the model achieved 91% accuracy for prediction of the labels in the validation set.

The project takes in various parameters and then provides the user with the most probable label for the respective inputs. Before training the model, we loaded the required datasets and performed necessary preprocessing steps. The feature extraction was done via Random Forest Classifier where we selected the top 25 features from the dataset. For training the LSTM model, we first reshaped the input shapes to pass it as an input to the model. We label encoded the values in the Label columns and then again to one-hot encoded format. The loss plot that we have calculated on the training and test dataset is to determine the model's training efficiency. The LSTM model that we designed takes in the parameters and trains upto the required number of epochs.

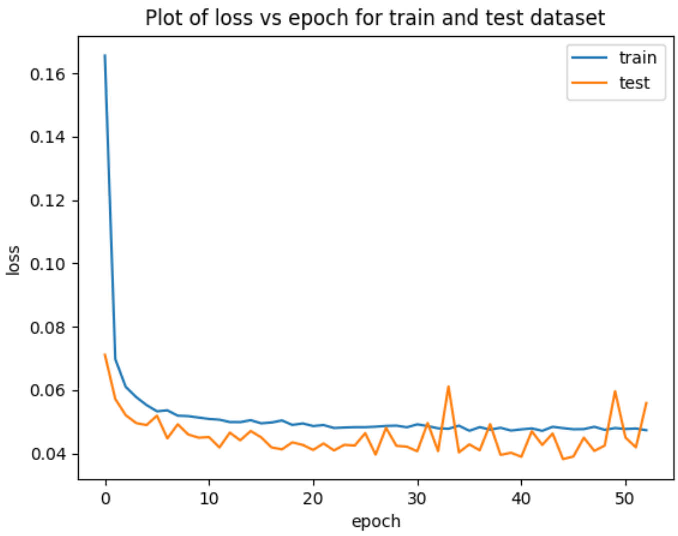

Figure 5.3: Loss plot for train and test dataset

CHAPTER 6
CONCLUSION

6.1 Conclusion

We have developed a Network Intrusion Detection System which is capable of detecting 91% of attacks in a computer network. The features were reduced from original 81 features to 25 features by using Random Forest Algorithm. We used the InSDN dataset for our assessments and obtained promising results.

6.2 Limitations

i. Our system can only detect DDoS, Normal and Probe traffics.
ii. Our system is only 91% accurate for real world traffic.

6.3 Future Enhancements

Additional dataset containing more variety of attacks can be added and used to train the model to increase the accuracy as well as to detect more variety of attacks.

CHAPTER 7

APPENDIX

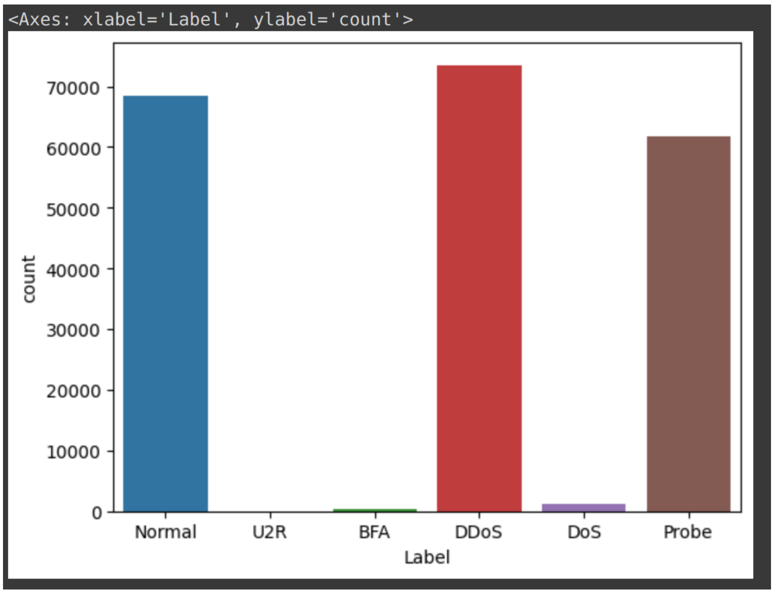

Figure 7.1: Initial Labels in the dataset

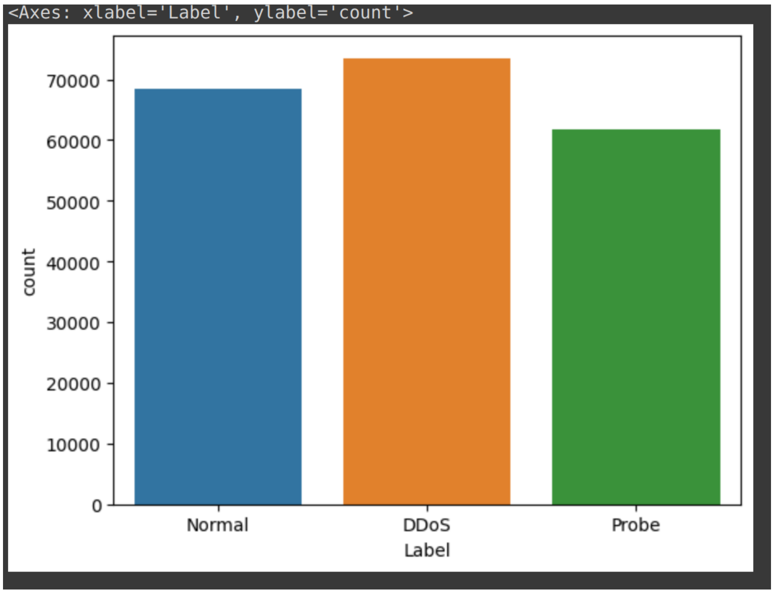

Figure 7.2: Final labels used for the project

	Top Features
0	Src_Port
1	Dst_Port
2	Protocol
3	Fwd_Pkt_Len_Mean
4	Bwd_Pkt_Len_Max
5	Bwd_Pkt_Len_Mean
6	Flow_Pkts_s
7	Bwd_IAT_Tot
8	Bwd_IAT_Mean
9	Bwd_IAT_Max
10	Bwd_IAT_Min
11	Bwd_Header_Len
12	Fwd_Pkts_s
13	Bwd_Pkts_s
14	Pkt_Len_Max
15	Pkt_Len_Mean
16	Pkt_Len_Std
17	Pkt_Len_Var
18	SYN_Flag_Cnt
19	Pkt_Size_Avg
20	Fwd_Seg_Size_Avg
21	Subflow_Fwd_Byts
22	Subflow_Bwd_Pkts
23	Subflow_Bwd_Byts
24	Init_Bwd_Win_Byts

Figure 7.3: Top 25 features selected from the dataset

Network Instrustion Detection System

Source IP	Destination IP	Source Port	Destination Port	Label
10.0.0.1	10.0.0.2	49094	80	DDOS
10.0.0.1	10.0.0.2	39515	80	DDOS
10.0.0.1	10.0.0.2	44871	80	DDOS
10.0.0.1	10.0.0.2	57774	80	DDOS
10.0.0.1	10.0.0.2	38950	80	DDOS
10.0.0.1	10.0.0.2	37869	80	DDOS
10.0.0.1	10.0.0.2	44871	80	DDOS
10.0.0.1	10.0.0.2	39515	80	DDOS
10.0.0.1	10.0.0.2	49298	80	DDOS
10.0.0.1	10.0.0.2	51468	80	DDOS
10.0.0.1	10.0.0.2	38950	80	DDOS
10.0.0.1	10.0.0.2	37489	80	DDOS
10.0.0.1	10.0.0.2	37949	80	DDOS
10.0.0.1	10.0.0.2	49094	80	DDOS
10.0.0.1	10.0.0.2	57774	80	DDOS
10.0.0.1	10.0.0.2	44871	80	Normal
10.0.0.1	10.0.0.2	51468	80	Normal
10.0.0.1	10.0.0.2	37489	80	Normal
10.0.0.1	10.0.0.2	37949	80	Normal
10.0.0.1	10.0.0.2	39515	80	Normal
10.0.0.1	10.0.0.2	37869	80	Normal
10.0.0.1	10.0.0.2	49298	80	Normal
10.0.0.1	10.0.0.2	38950	80	Normal
10.0.0.1	10.0.0.2	49094	80	Normal
10.0.0.1	10.0.0.2	57774	80	Normal
10.0.0.1	10.0.0.2	37489	80	Normal
10.0.0.1	10.0.0.2	39515	80	Normal
10.0.0.1	10.0.0.2	44871	80	Normal

Figure 7.4: Screenshot of the output

REFERENCES

[1] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *Ieee Access*, vol. 5, pp. 21 954–21 961, 2017.

[2] S. Al-Emadi, A. Al-Mohannadi, and F. Al-Senaid, "Using deep learning techniques for network intrusion detection," in *2020 IEEE international conference on informatics, IoT, and enabling technologies (ICIoT)*. IEEE, 2020, pp. 171–176.

[3] C. Xu, J. Shen, X. Du, and F. Zhang, "An intrusion detection system using a deep neural network with gated recurrent units," *IEEE Access*, vol. 6, pp. 48 697–48 707, 2018.

[4] M. S. E. Sayed, N.-A. Le-Khac, M. A. Azer, and A. D. Jurcut, "A flow-based anomaly detection approach with feature selection method against ddos attacks in sdns," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 4, pp. 1862–1880, 2022.

[5] R. A. Elsayed, R. A. Hamada, M. I. Abdalla, and S. A. Elsaid, "Securing iot and sdn systems using deep-learning based automatic intrusion detection," *Ain Shams Engineering Journal*, vol. 14, no. 10, p. 102211, 2023.

[6] M. Elsayed, N.-A. Le-Khac, and A. Jurcut, "Insdn: A novel sdn intrusion dataset," *IEEE Access*, 09 2020.