DETECTION OF DDOS ATTACK IN A CLOUD COMPUTING ENVIRONMENT USING DEEP LEARNING TECHNIQUE

BY

OSAWARU ALBERT

(PSC1813879)

DEPARTMENT OF COMPUTER SCIENCE

FACULTY OF PHYSICAL SCIENCE

UIVERSITY OF BENIN

BENIN CITY

JANUARY 2026

DETECTION OF DDOS ATTACK IN A CLOUD COMPUTING ENVIRONMENT USING DEEP LEARNING TECHNIQUE

BY

OSAWARU ALBERT

(PSC1813879)

A PROJECT REPORT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE, IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF BACHELOR OF SCIENCE (B.Sc) DEGREE IN COMPUTER SCIENCE OF THE UNIVERSITY OF BENIN

JANUARY 2026

APPROVAL

This project work is hereby approved by the department of computer science in partial fulfilment of the requirement for the award of Bachelor of Science (B.Sc) degree in Computer Science from the University of Benin

_____ _____

(Mrs) R.A. USIOBAIFO **Date**

Head of Department

CERTIFICATION

This is to certify that this project work was carried out by OSAWARU ALBERT, with matriculation number PSC1813879 under my supervision, it is adequate and satisfactory, both in scope and content, for the award of bachelor of science (B.Sc) degree in computer science of the University of Benin

_____ _____

Prof (Mrs) V.I Osubor **Date**

(Project Supervisor)

DEDICATION

I dedicate this project work to God Almighty. For giving me the strength and guidance to properly carry out and complete the work and also for his protection throughout my time at the University of Benin. This work is also dedicated to my parents, for making this journey as possible as easy they could, for encouraging me, and for guiding me

ACKNOWLEDGEMENT

I wish to express my profound gratitude to the following persons who contributed immensely in various ways towards the success of this research work and during my period in school.

Firstly, special thanks and credit to the Almighty God for His guidance, Protection, Wisdom, grace, understanding, strength and good health to fulfill my academic dreams.

My sincere appreciation goes to my project supervisor, Prof. (Mrs.) V. I. OSUBOR, for her patience, professional guidance and constant encouragement which greatly contributed to the success of this work. I am equally grateful to the Head of Department Dr. (Mrs.) A. R. Usiobaifo and all lecturers of the department of computer science, and indeed the entire staff and students of the University of Benin, Benin City, for the knowledge and academic support given to me during my programme.

I extend my heartfelt appreciation to my late father Chief John Osawaru A. Eribo, and my loving Mother (Mrs.) Patience Osawaru Ogbeide Eribo,

I will not fail to acknowledge the various authors of textbooks, Writers, Project and periodicals consulted during the course of writing this project

Finally, my humble thanks goes to everyone who contributed in one way or another to the success of this research but whose names are not mentioned, I sincerely say thank you.

Table of Contents

ABSTRACT

The security and reliability of cloud computing environments face significant threats from the escalating frequency and sophistication of Distributed Denial of Service (DDOS) attacks, which cause substantial financial losses and service disruptions while often serving as entry points for further system compromise. This research addresses this critical challenge by developing and evaluating deep learning-based detection models using two contemporary datasets: CICDDOS2019 (254,797 normal and 51,404 attack instances with 78 features) and IDS_ISCX_2012. To mitigate class imbalance, a balanced subset of 50,000 instances per class was created through random under-sampling, with optimal feature selection performed using the K-best method.

Two advanced recurrent neural network architectures were implemented and compared: Bidirectional Long Short-Term Memory (BI-LSTM) and Gated Recurrent Unit (GRU), both enhanced with temporal attention mechanisms to focus on critical attack patterns within sequential network traffic. Experimental results demonstrated that GRU outperformed BI-LSTM across both datasets, achieving accuracies of 0.93 and 0.65 on IDS_ISCX_2012 and CICDDOS2019 respectively, compared to BI-LSTM's 0.91 and 0.61. The GRU model's simplified architecture proved more computationally efficient while effectively addressing the vanishing gradient problem common in recurrent networks.

This study successfully establishes a robust framework for DDOS attack detection in cloud environments, contributing to enhanced network security through improved accuracy, reduced false positives, and practical implement ability for real-time threat mitigation.

CHAPTER ONE

Introduction

1.0 Background of the Study

The widespread adoption of cloud computing has revolutionized digital infrastructure by offering scalable, on-demand resources and services. However, this technological advancement has concurrently exposed critical security vulnerabilities, particularly to Distributed Denial of Service (DDOS) attacks. These attacks, which aim to disrupt service availability by overwhelming targets with malicious traffic, pose a significant threat to cloud environments due to their shared, elastic nature and expansive attack surface. As attackers employ increasingly sophisticated and multi-vector strategies, traditional detection mechanisms have become inadequate. Consequently, research has pivoted toward intelligent, data-driven systems powered by machine learning (ML) and deep learning (DL). This chapter systematically reviews the existing body of knowledge concerning cloud security, the taxonomy and evolution of DDOS attacks, and the progression of detection methodologies. It places special emphasis on contemporary deep learning techniques, including Bidirectional Long Short-Term Memory (BI-LSTM) and Gated Recurrent Units (GRU), which have shown remarkable efficacy in identifying complex attack patterns. Recent scholarly contributions underscore the pivotal role of advanced neural architectures in countering modern DDOS threats within cloud ecosystems.

1.1 Statement of the Problem

Despite the rapid adoption of cloud computing due to its cost-effectiveness and high availability, organizations face increasing security threats from sophisticated DDoS attacks. Traditional detection methods and basic machine learning approaches are insufficient—they produce high false alarms, lack accuracy, and cannot adapt to evolving attack patterns. There is a critical need

for an advanced, real-time deep learning-based detection system specifically designed for cloud environments to accurately identify DDoS attacks while minimizing false positives.

1.2 Aim and Objectives

The aim of this project is to classify DDOS attack using deep learning methods. The Specific Objectives are to:

1. Design a machine learning model for detecting Distributed Denial of Service in a Cloud Computing Environment using Bidirectional Long Short Term Memory (BI-LSTM) and Gated Recurrent Unit (GRU).
2. Implement the model designed.
3. Evaluate the performance of the implemented model in using F1 Score, Accuracy, Specificity, Sensitivity and Precision.

1.3 Significance of the study

The study is important for clients of cloud service in order to provide optimal usage of network resources. The proposed system will give a profoundly successful capacity of high recognition rate and low misleading problem proportion which is really difficult for existing DDoS guarded components. The DDoS attack detection system would analyze high-rates of network traffic in the cloud using deep learning algorithms. For optimal performance and reduced processing time, the recurrent neural network will be utilized to extract relevant network features, while the long short-time memory will be utilized to recognize DDoS attacks traffic from normal traffic.

1.4 Scope of the Study

The scope of the project is to:

1. To cover the utilization of deep learning strategies for the grouping of DDOS attack.
2. Developing model for detecting DDOS attack by recognizing all forms of attack accessible and grouping captured packets into normal attack or the sort of DDOS attack.

1.5 Methodology

For this study, two distinct cloud environments provided the traffic data. The IDS_ISCX_2012 dataset and the CICDDOS2019 data set. The data was cleaned by eliminating undesirable aspects such NAN values, port information, and feature selection after being stratified and organised to match the algorithm. A distinct set of DDOS attacks was used to handle each dataset separately.

The IDS_ISCX_2012 dataset has 2,381,532 normal traffic and 68,792 attack traffic with 8 features, but the CICDDOS2019 dataset has 254,797 normal traffic and 51,404 attack traffic each with 78 features. A balanced subset was produced by randomly choosing 50,000 instances from each class using best feature selection because there were more instances from the regular class than the attack class. By ensuring that the normal and attack classes were equally represented in the final subset, the random under-sampling strategy prevented bias towards the majority class and produced a more balanced dataset from the model training,

After processing, the data was sent to the deep learning algorithms BI-LSTM and GRU, which responded by tracking down the IP address of each data source that the algorithms had received. Data whose source IP address was blacklisted would be classified as attack traffic by BI-LSTM and GRU, which served as classifiers for the model. Data whose source IP address was not blacklisted would be classified as normal traffic by the two algorithms.

1.6 Term Definitions

Attack: Attacks are unlawful tasks completed on the computer network associated with a corporate organization. Network attacks are normally done by hackers, who uses machine learning to change, erase, or take private information. Aggressors as often as possible objective organization borders to get to interior frameworks.

Packets: As each piece of information being sent over a network is divided up into smaller components to allow for simple and speedy transmission over network links, they are the fundamental building blocks of information transfer over a network. Large amounts of information will be far more difficult to transport over a network without these fragmentations.

Vulnerability: A network weakness is a shortcoming or blemish in organization procedures, technical infrastructure, or software that, whenever took advantage of by an external threat, could prompt a security breach. Network weaknesses that are not physical usually include data or software.

Cloud: The significance of the expression "cloud" can be foggy, yet fundamentally, it refers to a huge network of servers, each serving a particular purpose. The cloud is not an actual object; rather, it is a sizable worldwide network servers that are interconnected and intended to work as a unified ecosystem.

Intrusion: Any unapproved activity on a digital network is referred to as an intrusion.Flooding: This happens when a router uses a no adaptive directing strategy to communicate an approaching packet to each outgoing link with the exclusion of the node on which the packet started.

Ping: Ping (otherwise called Bundle Web or Inter-Network Groper) is basic Web software that empowers clients to test and affirm whether a particular objective of IP address exists, and can acknowledge demands in computer network administration.

Ping of Death: A denial of-service (DDOS) attack that causes the freezing, destabilization, or crash of computer or services by besieging them with enormous information bundles is known as the "ping of death."

CHAPTER TWO

Literature Review

2.0 Introduction

The widespread adoption of cloud computing has revolutionized digital infrastructure by offering scalable, on-demand resources and services. However, this technological advancement has concurrently exposed critical security vulnerabilities, particularly to Distributed Denial of Service (DDoS) attacks. These attacks, which aim to disrupt service availability by overwhelming targets with malicious traffic, pose a significant threat to cloud environments due to their shared, elastic nature and expansive attack surface. As attackers employ increasingly sophisticated and multi-vector strategies, traditional detection mechanisms have become inadequate. Consequently, research has pivoted toward intelligent, data-driven systems powered by machine learning (ML) and deep learning (DL). This chapter systematically reviews the existing body of knowledge concerning cloud security, the taxonomy and evolution of DDoS attacks, and the progression of detection methodologies. It places special emphasis on contemporary deep learning techniques, including Bidirectional Long Short-Term Memory (BI-LSTM) and Gated Recurrent Units (GRU), which have shown remarkable efficacy in identifying complex attack patterns. Recent scholarly contributions underscore the pivotal role of advanced neural architectures in countering modern DDoS threats within cloud ecosystems.

2.1 Cloud Computing: Architecture and Inherent Security Challenges

Cloud computing delivers a suite of configurable computing resources—such as networks, servers, storage, applications, and services—over the internet with minimal management effort. It operates

primarily through three service models: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS), characterized by on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service (Mell & Grance, 2011).

Despite its transformative benefits, the cloud paradigm introduces unique and formidable security challenges that malicious actors frequently exploit:

Multi-tenancy and Isolation Failures: The shared physical infrastructure among multiple tenants (multi-tenancy) can lead to data leakage, side-channel attacks, and vulnerabilities within the hypervisor if isolation mechanisms are compromised (Ristenpart, Tromer, Shacham, & Savage, 2020).

- **Elasticity and Economic Denial of Sustainability (EDOS):** The very feature of auto-scaling can be weaponized. Attackers can trigger relentless resource provisioning during an attack, leading to exorbitant operational costs for the victim, a strategy known as EDOS.
- **Loss of Physical Control and Compliance Ambiguity:** Clients cede physical control of their infrastructure to cloud service providers (CSPs), creating dependencies and potential gaps in security responsibility and regulatory compliance.
- **Expanded Attack Surface:** The public accessibility and distributed architecture of cloud services significantly broaden the attack surface, making them susceptible to a wider array of network-based incursions.

These intrinsic vulnerabilities render cloud platforms attractive targets for DDoS attacks, which seek to compromise service availability by saturating resources with illegitimate traffic.

2.2 Taxonomy of Attacks on Computer Networks

Network attacks are deliberate actions intended to breach the confidentiality, integrity, or availability of computer systems and data. Within cloud environments, these attacks are increasingly sophisticated, often employing multiple vectors simultaneously.

Prominent categories of network attacks include:

1. **Malware Attacks:** Encompassing viruses, worms, trojans, and ransomware, these malicious software entities infect systems to steal data, gain unauthorized access, or disrupt operations.
2. **Phishing and Social Engineering:** Deceptive techniques that trick users into divulging sensitive information by masquerading as trustworthy entities.
3. **Man-in-the-Middle (MitM) Attacks:** The interception and potentially alteration of communication between two parties without their knowledge.
4. **SQL Injection:** An attack that inserts malicious SQL code into application queries to manipulate or destroy databases.
5. **Distributed Denial of Service (DDOS) Attacks:** Coordinated efforts from multiple compromised systems (a botnet) to flood a target with excessive traffic, rendering it inaccessible to legitimate users.

Among these, DDOS attacks represent a particularly acute threat to cloud availability due to their scalability, the ease of orchestrating them via vast botnets (often comprising IOT devices), and their potential for causing severe financial and reputational damage.

2.3 Taxonomy of Attacks on Computer Networks

Network attacks are deliberate actions intended to breach the confidentiality, integrity, or availability of computer systems and data. Within cloud environments, these attacks are increasingly sophisticated, often employing multiple vectors simultaneously.

Prominent categories of network attacks include:

- **Malware Attacks:** Encompassing viruses, worms, trojans, and ransomware, these malicious software entities infect systems to steal data, gain unauthorized access, or disrupt operations.
- **Phishing and Social Engineering:** Deceptive techniques that trick users into divulging sensitive information by masquerading as trustworthy entities.
- **Man-in-the-Middle (MitM) Attacks:** The interception and potentially alteration of communication between two parties without their knowledge.
- **SQL Injection:** An attack that inserts malicious SQL code into application queries to manipulate or destroy databases.
- **Distributed Denial of Service (DDoS) Attacks:** Coordinated efforts from multiple compromised systems (a botnet) to flood a target with excessive traffic, rendering it inaccessible to legitimate users.

Among these, DDOS attacks represent a particularly acute threat to cloud availability due to their scalability, the ease of orchestrating them via vast botnets (often comprising IOT devices), and their potential for causing severe financial and reputational damage.

2.4 Overview and Evolution of DDoS Attacks

A Distributed Denial of Service (DDOS) attack is a malicious campaign designed to disrupt the normal traffic flow of a targeted server, service, or network by inundating it with a deluge of internet traffic. Distinguished from simpler Denial-of-Service (DOS) attacks that originate from a single source, DDoS attacks leverage numerous geographically dispersed, compromised systems to generate attack traffic, forming what is known as a botnet.

Salient characteristics of modern DDOS attacks include:

1. **Distributed Origin:** Attack traffic emanates from a multitude of sources, complicating attribution and mitigation.
2. **Amplification Techniques:** Exploitation of protocols with asymmetric responses (e.g., DNS, NTP, Memcached) to magnify attack volume, turning a small query into a massive traffic surge directed at the victim.
3. **Botnet Utilization:** The harnessing of compromised Internet of Things (IOT) devices, servers, and personal computers, often unbeknownst to their owners, to form powerful attack networks.
4. **Increasing Sophistication:** Contemporary attacks employ tactics such as traffic encryption, "low-and-slow" attack vectors that fly under detection radars, and multi-vector approaches that combine several attack types to evade defensive measures.

The impact of a successful DDoS attack in the cloud extends far beyond temporary service disruption. It can lead to substantial financial losses from exhausted resources, violations of Service Level Agreements (SLAs), and long-term erosion of customer trust and brand reputation.

2.5 Classification of DDoS Attack Types

DDoS attacks are typically categorized based on the layer of the Open Systems Interconnection (OSI) model they target and their operational methodology.

2.5.1 Volumetric Attacks (Targeting Network/Transport Layers 3 & 4)

These attacks aim to consume all available bandwidth of the target by generating massive volumes of traffic.

- **UDP Floods:** The target is bombarded with User Datagram Protocol (UDP) packets directed at random ports.
- **ICMP Floods:** The victim is overwhelmed with Internet Control Message Protocol (ICMP) Echo Request (ping) packets.
- **Amplification/Reflection Attacks:** Attackers spoof the victim's IP address and send small queries to publicly accessible servers (like DNS or NTP servers) that respond with much larger replies, all directed at the target.

2.5.2 Protocol Attacks (Targeting Network/Transport Layers 3 & 4)

These attacks exploit weaknesses in network protocols to exhaust connection state tables or other server resources.

1. **SYN Floods:** The attacker sends a rapid succession of TCP connection requests (SYN packets) but never completes the three-way handshake, leaving the target with a backlog of half-open connections.

2. **Ping of Death:** The transmission of malformed or oversized packets that can crash or destabilize the target system.
3. **Slow loris:** A low-and-slow attack that opens many connections to a web server and keeps them open for as long as possible by sending partial HTTP requests.

4. **2.5.3 Application-Layer Attacks (Targeting Layer 7)**

These attacks target specific applications or services and often mimic legitimate user behavior, making them harder to detect.

1. **HTTP/HTTPS Floods:** Overwhelming a web server with a high rate of seemingly valid HTTP GET or POST requests.
2. **DNS Query Floods:** Sending an excessive number of DNS queries to exhaust the resources of a DNS resolver.
3. **SSL/TLS Exhaustion:** Forcing a server to perform the computationally expensive process of negotiating SSL/TLS handshakes repeatedly (Zhang et al., 2023).

A concerning contemporary trend is the rise of **multi-vector DDOS attacks**, which combine two or more of the above attack types simultaneously to overwhelm different parts of an infrastructure, thereby complicating detection and mitigation efforts (Patel & Joshi, 2022).

2.6 Fundamentals of Deep Learning

Deep learning, a powerful subfield of machine learning, utilizes artificial neural networks with multiple layers (hence "deep") to learn representations of data with multiple levels of abstraction. A key advantage over traditional ML is its ability to automatically discover the representations

needed for feature detection or classification from raw data, reducing the need for manual feature engineering.

2.6.1 Core Neural Network Concepts

Neural networks consist of interconnected nodes or "neurons" organized in layers: an input layer, one or more hidden layers, and an output layer. Each connection between neurons has an associated weight. During training, these weights are adjusted through an optimization process called back propagation to minimize the difference between the network's predictions and the actual targets.

2.6.2 Deep Learning Architectures for Sequential Data Analysis

For the task of DDoS detection, which involves analyzing time-ordered sequences of network traffic (packets or flows), specific deep learning architectures are particularly well-suited:

1. **Recurrent Neural Networks (RNNs):** Specifically designed for sequential data, RNNs have internal memory (hidden state) that captures information about previous inputs in the sequence. However, standard RNNs struggle with learning long-term dependencies due to the vanishing gradient problem.
2. **Long Short-Term Memory (LSTM):** An advanced type of RNN that incorporates a gating mechanism (input, forget, and output gates) to regulate the flow of information. This architecture effectively mitigates the vanishing gradient problem, making it capable of learning long-range dependencies in sequences—a critical feature for identifying attack patterns spread over time.

3. **Bidirectional LSTM (BI-LSTM):** An extension of LSTM that processes input sequences in both forward and backward directions. This allows the network to capture contextual information from both past and future states relative to any point in the sequence, providing a richer understanding of traffic patterns and often yielding superior performance in detection tasks.
4. **Gated Recurrent Unit (GRU):** A variation of the LSTM that simplifies the architecture by combining the input and forget gates into a single "update gate" and having a "reset gate." This results in fewer parameters, which typically translates to faster training times and lower computational complexity while often maintaining comparable performance to LSTMs, making GRUs attractive for real-time applications.
5. **Convolutional Neural Networks (CNNs):** Although predominantly used for spatial data (like images), one-dimensional CNNs (1D-CNNs) can be effectively applied to sequential data to extract local, translation-invariant patterns. They are frequently used in hybrid models alongside RNNs for DDoS detection.

2.10 Review of different strategies for identifying ddos attacks.

This segment stresses the continuous DDoS detection techniques, and algorithms proposed and applied reliant upon the distributed work.

As indicated by Hameed and Ali (2018), a design called HADEC was familiar with recognizing live high-rate DDoS attack which occurs at system and application layers, for instance, TCP-SYN, HTTP GET, UDP, and ICMP. The design is extracted from two principle parts: recognize server and catch server. Live DDoS identification begins by getting the server that is proficient to get live framework traffic and move the log to the area server for planning. The discovery computers approach packets for UDP, ICMP, and HTTP to classify an attack if the source affiliation surpasses

the pre-portrayed limit. The proposed area gives an easy answer with any budgetary establishment, just as little as the medium associations.

Behal *et al*. (2018) proposed a recognizable proof strategy called D-FACE to recognize four traffic types: genuine client, low-rate, high-rate, and glimmer occasion traffic. The identification uses entropy differentiation that contains a common traffic stream, while the assessment of source IP entropy is the location network to figure out the attack. The acknowledgement begins with the extraction of the connected header that groups the system into an exceptional system stream. The seclusion of low-rate, high-rate, and blaze occasion traffic is based on differentiating the current moving toward traffic rate in each time window and dependent information traffic regard.

A DDoS detection technique was proposed by Singh *et al*. (2018), to identify HTTP DDoS attacks through Machine Learning to deal with recognising botnets from genuine clients in distinctive attack traffic, genuine traffic, and glimmer traffic. The proposed structure is conveyed as a mediator and assesses the client lead instead of noticing the entire traffic. The proposed work perceives the botnet source and investigates client conduct to identify vindictive requests against the web server.

Sreeram and Vuppala (2017) proposed Machine Learning matrices with a bio-propelled bat estimation to allow fast and early distinguishing proof of HTTP DDoS attacks. The work intertwined time breaks, rather than client sessions and bundle examples to create a discovery calculation. The time break utilizes AI lattice by designating an estimation of the most extreme sessions for one-time intervals and figuring different sessions in a one-time break to recognize DDoS attacks at application layers. The grid furthermore addresses two pages of HTTP GET

demand. The reoccurrence of a site page got to by the clients and a time whole between the first page requesting and the second page resolving the screen client conduct.

Aborujilah and Musa (2017) introduced a cloud-based area of HTTP DDoS by using the measurable approach with the statistical measure structure. The identification introduced two computations known as planning and testing to see an alternate sort of HTTP flooding attack subject to attack conduct. A planning calculation was used to decide the types of traffic sort, and the testing computation was used to conclude the sorts of traffic got. The outcomes acquired from this investigation had been assessed by using the disarray structure to measure recognition execution and give results of interior and external cloud conditions.

Singh and De (2017) used multi-facet perception with an inherited calculation (MLP-GA) to distinguish HTTP DDoS attacks. The proposed area utilized four parameters to produce identification into application layers. An average client has express interim, as a genuine client look and scrutinizes while getting to a site page and while moving to various pages. The identification technique suggested by the analysts checks the quantity of HTTP GET requests got by the web server and discovers the quantity of Internet Protocol addresses concentrating on the server over 20 seconds. The proposed area likewise examines the port number used by HTTP DDoS, as ports used by HTTP DDoS attackers are changed and remain open. The discovery procedure used fixed edge length to lead acknowledgement, as shown by these analysts, HTTP DDoS attackers use static assembly lengths.

Hoque *et al*. (2017) proposed a strategy for the recognition of DDoS at the application layer constantly at the lamentable setback end. The proposed work utilized software and hardware gotten from the system made to recognize normal traffic from fake traffic. The three essential segments

consolidated into the framework were the pre-processor, hardware module, and security supervisor, which arranged source IPs, IPs source variety file, and bundle to distinguish the attack.

2.11 OVERVIEW OF BI-DIRECTIONAL LONG SHORT-TIME MEMORY

One of the greatest mishaps of the RNN is that it dials back as the hole between the recently handled information and the place of prerequisite increments. This is the benefit the LSTM has over other RNN models (Kumar, 2018). They are planned so that they can get away from the drawn-out reliance issue of the other RNNs. An average LSTM has four layers with each layer corresponding to one another. The figure below describes the four repeating modules of an LSTM.

The LSTM has three gate activation functions which include σ1, σ2, and σ3 as displayed in Figure 2.2 above. It additionally has two output activation functions. The image π and Σ address component-wise duplication and expansion respectively. The link activity is addressed by the image (•) bullet. The major part of LSTMs is cell state, a line running from Memory from the Past Block (St−1) to Memory from the Current Block (St). It permits the information to flow straight down the line. The network can choose how much previous information to flow. It is controlled through the first layer (σ1).

For an LSTM organization to recall data for longer periods in the default conduct of the LSTM. LSTM networks have a comparative design to the RNN, yet the memory module or rehashing module has an alternate LSTM. The block graph of the rehashing module will seem to be the picture underneath.

CHAPTER THREE

3.1 Methodology

This project research aims to use two different datasets and two different algorithms or techniques to ensure better accuracy. The research work implemented a Distributed Denial of Service on Gated Recurrent Unit (GRU) and Bi–directional Long Short-Term Memory (BI- LSTM). BI-LSTM, an advanced form of LSTM is one of the types of Recurrent Neural Network (RNN). It is commonly used where sequence-to-sequence actions are needed allowing the neural network to process data in both forward (past to future) and reverse (future to past) directions. This type of network can be utilized in text classification, speech recognition, and forecasting models. In a situation where steps of the input sequence are available, BI-LSTMs train two LSTMs on the input sequence instead of one.

This research work also integrated a Gated Recurrent Unit (GRU), which is an improvement over the standard RNN, it was introduced by Kyunghyun Cho *et al* in the year 2014. The GRU is one of the particular model of RNN that aims to execute the machine learning tasks relating to memory and clustering using connections from the sequences of nodes, for example, in speech recognition. One of the common problem with RNN is vanishing gradient problem, GRU can help solve this problem by adjusting the input weights of the neural network. The major contrast between GRU and LSTM is that GRU operates with two gates which are; changed and update, while LSTM operates with three gates which are; input, output, and forget gates. GRU resembled LSTM but has an easier architecture, using gates to control the flow of information. If the dataset is small, then GRU is preferred otherwise LSTM is for the larger dataset.

3.2 System Architecture

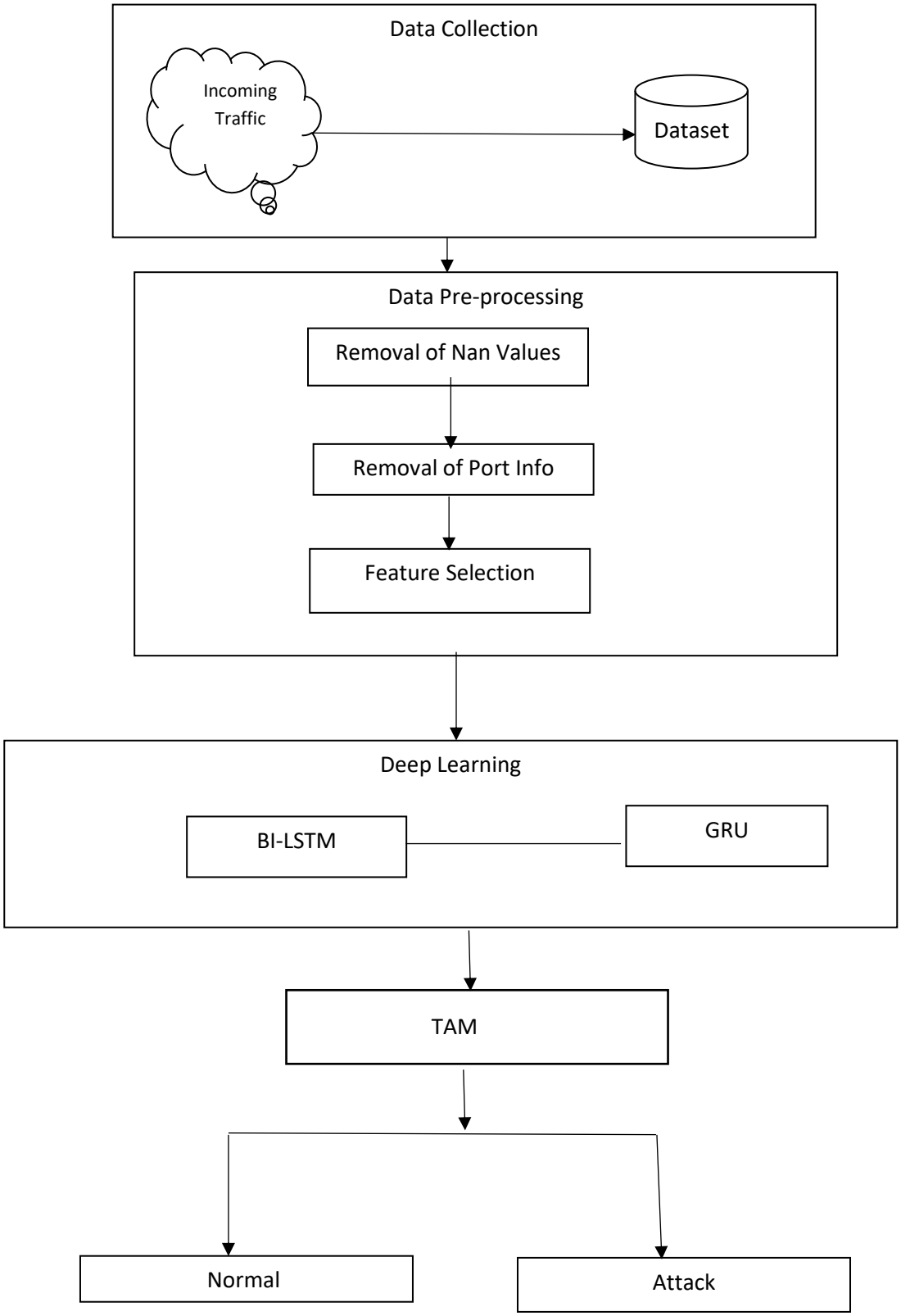

Figure 3.1: The System Architecture

Figure 3.1 represent the architecture of the system. There are five major sections of this system. They include the following:

a. Data Collection: This section is designated for the collection of data from different cloud-based sources. The data collected are stratified and structured to suit the developed model

b. Data Preparation: The data are being cleaned in this phase. Removal of all unnecessary data in the datasets.

c. Deep Learning: The result of the data preparation phase is received in this phase and is used to determine the system choice. Each of the deep learning models receives the same data and this data are then processed using the algorithm and structure of the model and then their output is collected and sent into the final model.

d. Temporal Attention Mechanism : This was ntroduced to allows the model to dynamically weigh the importance of different time steps within the sequential network traffic data by assigning higher attention weights to more informative time steps, the model will focus on relevant patterns and capture crucial temporal dependences.

3.3 Datasets

Two datasets will be utilize to train and test the model. The result of the dataset will be compared against the two algorithms to determine the most effective among the two datasets. The datasets are IDS_ISCX_2012 and CICDDOS2019.

The IDS_ISCX_2012 dataset was originally collected for the purpose the general intrusion detection. However, later versions were DDoS included in the goals and subsequently streamlined for DDoS detection. The challenges of most public datasets available at the time which prompted

the institute to create and processed a public dataset with the sole aim of intrusion detection include the following:

1. Scarcity of Public Datasets.
2. Privacy of the data owner and data proprietor is not willing to share based on the nature of the datasets.
3. Heavy anonymity in the dataset and dataset may not reflect the present-day trend.

In other for the data proprietor to generate a dataset that will solve the underlying challenges, a step-by-step approach is established to address this need. The datasets contain a great deal of description of the data. Real traces are analyzed to ensure that profile agents are created in such a way that the agents that generate traffic like HTTP, SMTP, SSH, IMAP, POP3, and FTP are represented. For user behavioural simulations, the proprietary institute used the institute an abstracted profile of the user in the institute.

The datasets possess the following characteristics:

1 Realistic network and traffic: Unintended properties should not be demonstrated by a dataset in both network and traffic-wise. This is to establish a true representation of the actual effects of attacks over the network and the corresponding feedback of workstations. Because of this, the traffic needs to appear and behave as realistically as possible. Attack and normal traffic inclusive. The raw data will be affected negatively by any artificial post-capture trace insertion. This may bring possible inconsistencies in the final dataset. Therefore, any such adjustments are not recommended.

2 Labeled dataset: A labelled dataset carries out huge benefits in the evaluation of various detection mechanisms. For this reason, creating a dataset in a controlled and deterministic

environment allow for differentiating abnormal activity from normal traffic; thus, erasing the impractical process of manual labelling.

3 Total interaction capture: The amount of information available to detection mechanisms are of very essential importance as this provides the means to detect abnormal behaviour. In other words, this information is very necessary for post-evaluation and the right explanation of the results. Thus, it is a vital demand for a dataset to include all network interactions, either within or between internal Local Area Networks (LANs).

4 Complete capture: Privacy concerns related to sharing real network traces have been one of the major challenges for network security researchers as data providers are often unwilling to divulge such information. Consequently, most such traces are either used within which limits other researchers from precisely evaluating and comparing their systems, or are heavily unidentified with the payload entirely removed resulting in reduced utility to researchers. In this work, the main objective is to produce network traces in a managed tested environment, thus removing the need for any sanitization and thereby protecting the naturalness of the resulting dataset.

5 Diverse intrusion scenarios: In recent years, attacks have increased in occurrence, size, variety, and complexity. The range of threats has also changed into more hard schemes, this includes service and application-targeted attacks. Such attacks can cause far more consequential disruptions than traditional brute force attempts and also require a more in-depth perception of IP services and applications for their detection. This objective aims to perform various sets of multistage attacks; each meticulously crafted and directed towards recent trends in security threats. This objective frequently labels many of the available datasets as not effective and unfit for evaluating research results. They are generated using two types of user profiles and multistage attacks, such as Heart bleed, and a variety of DoS and DDoS attacks. CICFlowMeter is used to pre-process the collected

traffic. It has thirty (30) network traffic elements, working towards generating varied DoS and DDoS traffic data. The dataset used consists of details of over 100,000 packets which enlist both normal traffic and DDOS traffic. The datasets used have 30 elements and the last column is used for the classification into "normal" and "attack". The dataset used consists of 45,665 benign traffic types and DDOS attacks.

The CICDDOS2019 dataset contains benign and the most up-to-date common DDoS attacks, which resemble true real-world data (PCAPs). It also includes the results of the network traffic analysis using CICFlowMeter-V3 with labelled flows based on the time stamp, source, and destination IPs, source and destination ports, protocols, and attack (CSV files).

The dataset contains many different modern reflective DDoS attacks such as PortMap, NetBIOS, LDAP, MSSQL, UDP, UDP-Lag, SYN, NTP, DNS, and SNMP. Attacks were executed afterwards. The datasets were collected using twelve (12) forms of Distributed Denial of Service attacks including NTP, DNS, LDAP, MSSQL, NetBIOS, SNMP, SSDP, UDP, UDP-Lag, WebDDoS, SYN, and TFTP, and seven (7) attacks including PortScan, NetBIOS, LDAP, MSSQL, UDP, UDP-Lag and SYN in the testing.

3.4.1 System Flowchart

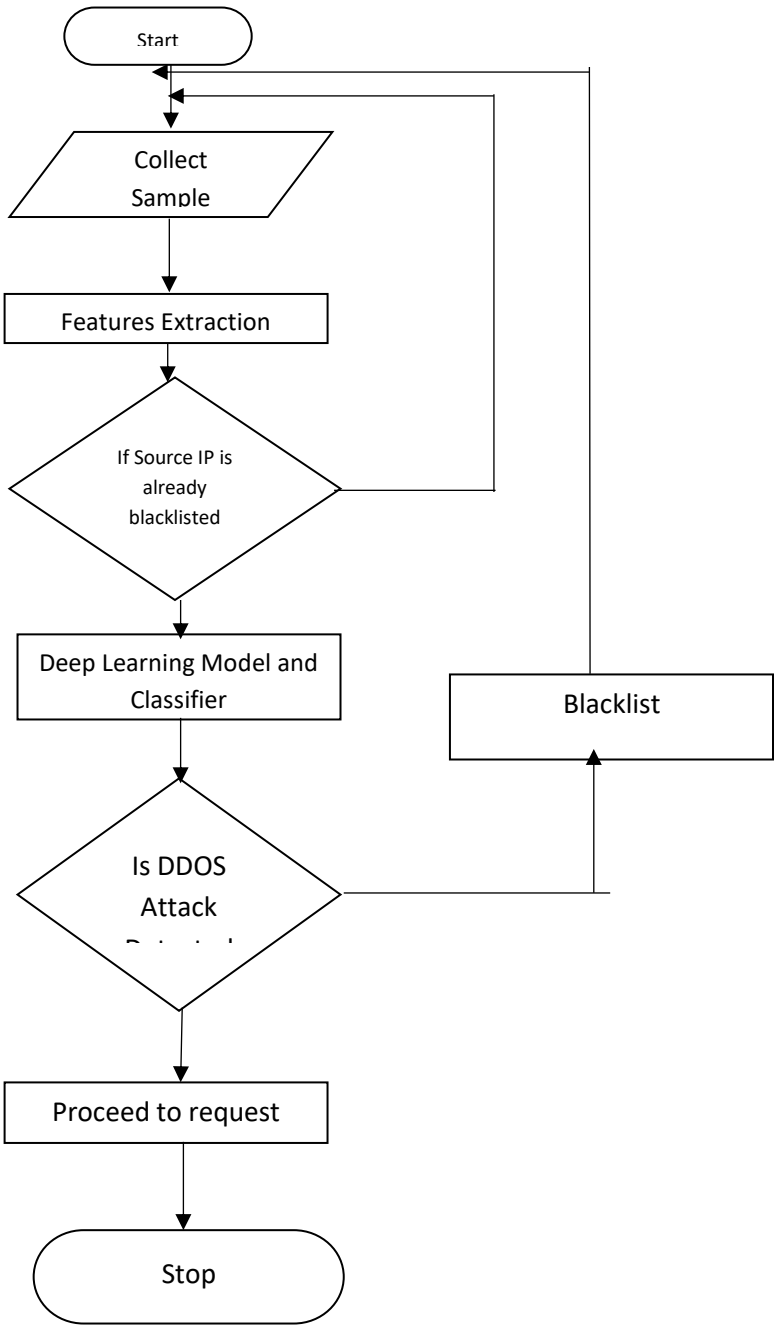

Figure 3.2: System Flowchart.

Figure 3.2 begins the proposed system by taking input from the collected packets and then starting the method involved with gathering the significant highlights from every parcel information that has been gathered. The source IP of the model is checked if it has been previously blacklisted by

the system. By blacklist, it implies the IP has prior been thought to be for an attack prior so it is being disconnected. If the parcels go liberated from doubt, they will be passed to the BI-LSTM Model before being pre-trained and afterwards to the classification are then anticipated by the model. If the model returns a negative or 0 which suggests an attack, the packet is confined and the extraction of the packets is to determine the type of source it's coming from and its classification. Below is a chart showing the proposed system block diagram.

4.2 System Model Flow Chart

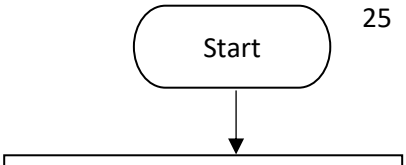

Figure 3.3: System Model Flow Chart

According to Figure 3.3, the training processes begin with the collection of the dataset. The dataset preparation begins with balancing the datasets and ensuring that the dataset is not imbalanced. The next phase is the feature extraction phase in which all the important features are selected. The

layers are then predefined with each of the layers being segmented. The model has 4 layers. The components of these layers are discussed in the next sections.

3.5 System Algorithm

The system algorithm will work as following:

Capture packets sent to cloud service

Input packets data's into system

Feature Extraction of the packet data

Check if IP has been blacklist

If IP has previously being blacklisted

    Block Request

Else

    Continue with procedure

Pass The Result to the pre-trained Deep Learning model

If result is Normal attack

    Continue to request

Else

    Black List IP

Request

    Stop

3.5.1 BI-LSTM Model Algorithm

Inputs: Datasets

Size ← length (series) * 0.70

Train← series [0...size]

Test ← series [size...length (size)]

//Set the random seed to a fixed value

Set random seed (7)

set option = "B" (BiLSTM) // Fit a BiLSTM model to training data Procedure fit_lstm_Bilstm(train, epoch, neurons, option)

X ← train 7. y ← train - X

model = Sequential()

if option = 'L':

Model.add(LSTM(neurons), stateful=True))

else option = 'B':

model.add(Bidirectional(LSTM(neurons, stateful=True))

model.compile(loss='mean_squared_error', optimizer='adam')

for each in range(epoch) do

model.fit(X, y, epochs=1, shuffle=False)

model.reset_states()

end for return model # Make a one-step forecast Procedure forecast-lstm(model, X)

yhat ← model.predict(X) return yhat

epoch ← 1

neurons ← 4

predictions ← empty

lstm_model = fit_lstm_Bilstm(trmachine Learning n,epoch, neurons, option)

//Forecast the training dataset

lstm_model.predict(train)

//Walk-forward validation on the test data

for each i in range(length(test)) do

// make one-step forecast

X ← test[i]

yhat ← forecast_lstm(lstm_model, X)

// record forecast

predictions.append(yhat)

expected ← test[i]

end for

MSE ← mean_squared_error(expected, predictions)

Return (RMSE ← sqrt(MSE))

3.5.2 Bidirectional LSTM (BLSTM) Mathematical Model

To store temporal information, LSTM was built using specific memory cells. Because of its structure, LSTM is better than traditional recurrent neural networks at remembering distant features. The mathematical model of LSTM is expressed below

$$f_i^l = \sigma\left(W_{(f)}^l \vec{h}_i^{l-1} + V_{(f)}^l \vec{h}_{i-1}^l + b_{(f)}^l\right), \quad (1)$$

$$i_i^l = \sigma\left(W_{(i)}^l \vec{h}_i^{l-1} + V_{(i)}^l \vec{h}_{i-1}^l + b_{(i)}^l\right), \quad (2)$$

$$o_i^l = \sigma\left(W_{(0)}^l \vec{h}_i^{l-1} + V_{(o)}^l \vec{h}_{i-1}^l + b_{(o)}^l\right), \quad (3)$$

$$g_i^l = \sigma\left(W_{(g)}^l \vec{h}_i^{l-1} + V_{(g)}^l \vec{h}_{i-1}^l + b_{(g)}^l\right), \quad (4)$$

$C_i^l = f_i^l \Theta C_{i-1}^l + i_i^l \Theta g_i^l,$ (5)

$h_i^{\vec{l}} = o_i^l \Theta \tanh(C_i^l),$ (6)

Where the parameters used were defined as below;

f_i^l = input gate

i_i^l = forget gate

o_i^l = output gate

g_i^l = candidate gate

C_i^l = cell gate

$h_i^{\vec{l}}$ = hidden gate

W^l = weight matrices between cells of layer (l−1)–*l*

V^l = weight matrices between consecutive cells of layer *l*

b^l = the bias vector at each layer

σ = The sigmoid function

Θ = The element-wise multiplication

BLSTM process data in a backward and forward directions by using two separate layers of LSTM. By using the equation (6) to calculate the forward hidden state $h_i^{\vec{l}}$ and calculating the backward hidden state $h_i^{\overleftarrow{l}}$ the same way as backward hiding state and integrate together and then push forward to the layer that follow it.

$h_i^{\leftrightarrow l} = [\frac{h_i^{\vec{l}}}{h_i^{\overleftarrow{l}}}]$ (7)

By using information from both directions, the BDLSTM is more effective at obtaining the relationships between the elements in a whole sequence than the traditional LSTM, which only remembers the characteristics in one direction.

3.5.3 THE MATHEMATICAL MODEL OF GATED RECURRENT UNIT.

The mathematical model of the Gated Recurrent Units is expressed below

$R_t = \sigma\ (X_t W_{xr} + H_{t-1} W_{hr} + b_r)$ (1)

$Z_t = \sigma\ (X_t W_{xz} + H_{t-1} W_{hz} + b_z)$ (2)

$\widetilde{H_t} = \tanh(X_t W_{xh} + (R_t \odot H_{t-1}) W_{hh} + b_h)$ (3)

$H_t = Z_t \odot H_{t-1} + (1 - Z_t) \odot + \widetilde{H_t}$ (4)

Where,

R_t = reset gate

Z_t = update gate

X_t = input

t = time

σ = The sigmoid function

b_r, b_z, b_h = biased parameters

W_{xr}, W_{xz}, W_{hr}, W_{hz}, W_{xh}, W_{hh} = weight parameters

H_{t-1} = old candidate state

Θ = The element-wise multiplication

$\widetilde{H_t}$ = new candidate state

The element-wise convex combinations of the old candidate state and the new candidate state gives the updated equation for Gated Recurrent Units.

3.5.4 Temporal Attention Mechanism

Temporal Attention Mechanism is a type of attention mechanism commonly used in sequential models, such as recurrent neural networks (RNNs), to capture temporal dependencies in sequences. It allows the model to selectively focus on specific parts of the input sequence that are most relevant for the current prediction or decision.

Algorithm:

1. Input: The input to the Temporal Attention Mechanism consists of two tensors: the query tensor Q and the value tensor V.

2. Encoding: the query tensor Q is processed through a neural network layer (such as a Dense layer) to obtain a transformed query tensor Q', which matches the dimensions of the value tensor V.

3. Alignment Scores: The alignment scores are computed by taking the dot product between the transformed query tensor Q' and each element in the value tensor V. This step measures the relevance or similarity between the query and each element in the sequence.

4. Attention Weights: The alignment scores are often passed through a softmax function to obtain attention weights. The softmax function normalizes the scores, ensuring they sum up to 1, representing the importance or weight assigned to each element in the sequence.

5. Weighted Sum: The attention weights are multiplied element-wise with the value tensor V. This operation gives higher importance to elements in the sequence that have higher attention weights.

6. Context Vector: The weighted sum of the value tensor V produces a context vector, which is a representation that captures the relevant information from the input sequence.

7. Output: The context vector is then used as input to subsequent layers or directly used for making predictions.

3.5.5 Mathematical Model:

Let's denote the query tensor as Q with dimensions (batch_size, query_length, query_dim), and the value tensor as V with dimensions (batch_size, value_length, value_dim).

Encoding: Typically, a transformation is applied to the query tensor Q to match the dimensions of the value tensor V:

Q' = Transformation (Q) with dimensions (batch_size, query_length, value_dim)

Alignment Scores: The alignment scores are calculated by taking the dot product between the transformed query tensor Q' and the value tensor V:

alignment_scores = Q' dot V^T with dimensions (batch_size, query_length, value_length)

Attention Weights: The alignment scores are often passed through a softmax function to obtain attention weights:

attention_weights = softmax(alignment_scores) with dimensions (batch_size, query_length, value_length)

Weighted Sum: The attention weights are multiplied element-wise with the value tensor V:

weighted_sum = attention_weights dot V with dimensions (batch_size, query_length, value_dim)

Context Vector: The weighted sum operation produces a context vector that captures the relevant information from the input sequence:

context_vector = sum(weighted_sum, axis=2) with dimensions (batch_size, query_length, value_dim)

The context vector is then used as input to subsequent layers or for making predictions, depending on the specific task.

The Temporal Attention Mechanism allows the model to focus on different parts of the input sequence dynamically, providing flexibility to capture long-term dependencies and improve performance in sequential tasks such as machine translation, text summarization, and speech recognition.

3.6 Deep Learning Model

The construction of the model is consequently depicted. The planned profound learning model incorporates the BI-LSTM model and GRU model. The models were created utilizing 100,000 example traffic chosen from the dataset (CICDDOS2019 and IDS_ISCX_2012 datasets), with 50,000 being typical traffic and 50,000 being attack traffic. The dataset has a state of (100,000,25). The following is a table showing the boundaries utilized for the plan of the BI-LSTM Model.

Table 3.1: BI-LSTM Model Table.

The model above utilizes four significant layers, every one of the layers has its novel initiation capability to frame its grouping of hubs. The enactment capability utilized in every one of the capabilities incorporates the tanh likewise alluded to as the exaggerated digression actuation capability. The exaggerated digression actuation capability accepts any genuine worth as info and result values in the reach - 1 to 1. The bigger the input (more sure), the nearer the result worth will be to 1.0, though the more modest the info (worse), the nearer the result will be to - 1.0. The model additionally carries out relu the corrected linear activation. The rectified linear activation enactment capability or ReLU for short is a piece of insightful linear capability which will yield the input directly if it is positive, or else, it will yield zero.

Number of Layers	4 Layers: Input layers, BI-LSTM, Dense Layer and Output Layer
Activation Function	Tanh For Bi- LSTM layer, Relu and Sigmoid
Optimizer	Adam
Loss Function	Binary Cross Entropy
Dropout	0.2
Kernel Regularize	12
Number of Epochs	200
Verbose	1

The table beneath shows the boundary/organisation utilized for the plan of the GRU (Gated Recurrent Unit)

Table 3.2 Gated Recurrent Unit Model Table.

Number of Layers	4 layers: Input layer, GRU, Dense Layer and Output layer
Activation Function	Tanh for GRU Layer, Relu and Sigmoid
Recurrent Activation	Sigmiod
Bias	True with 0 as initializer value
Recurrent Initializer	Orthogonal
Recurrent Regular	None
Optimizer	Adam
Loss Function	Binary Cross Entropy
Dropout	0.2
Kernel Regularize	12
Number of Epochs	200
Verbose	1

CHAPTER FOUR

IMPLEMENTATION, RESULTS AND DISCUSSIONS

4.1 Implementation

This part depicts the execution, result, and evaluation of the system designed; it involves the testing of the framework, project plan, project executives, risk the board, quality administration, and the different contemplations taken while fostering the framework.

System execution is the change of steps of the framework plan to genuine directions and codes shown to play out the assignments demonstrated in the framework in particular. The target of this research is to develop a cloud-based detection of a distributed denial of service attack that will empower us to decide the different kinds of traffic (either attack or normal traffic) and the wellspring of the traffic. The framework planned executed the GRU and LSTM methods towards grouping DDOS in a cloud-based network. To execute the created framework, there are essentially two significant classes of prerequisites which include: Hardware necessity and Software Prerequisite

4.1.1 Hardware Prerequisite

The hardware requirement includes all physical components required for the model to perform optimally in a test environment. The minimum hardware configuration required to be able to implement the model includes the following:

a. Hard Circle (HDD): 500GB

b. Slam: 8Gb Smash

c. Illustrations Card: NVidia Quadro with Cuda

d. Processor: Center i7

e. Network Connection point Card

4.1.2 Software Requirement

The minimum hardware configuration required to be able to implement the model includes the following:

a. Python 3.7

b. Working Framework - Windows 10

c. Boa constrictor 1.9.1.2

d. Jupyter Scratchpad 6.03

e. Tensorflow

f. Pandas

g. Matplotlib

h. PyDot

i. Numpy

j. Seaborn

k. Scikit Learn

l. Keras

4.2 Platform, Language, and Tools Used

The essential programming language utilized in the development of the model is the Python Programming language. Python is a universally useful high-level programming language widely used today. It is an elucidate language that has a design pattern that renders readability of code and syntax structure which allows the programmers to communicate a complicated idea in a couple of lines of code through builds expected to empower composing clear projects on both a little and huge scope. Also, it's an object-oriented programming and organized programming language which has a significant element called dynamic name goal or late restricting which binds methods and variable names during program execution.

Coming up next is the benefit that Python offers as a programming language:

a. Object Oriented: In Python, everything is an object. Python can be effortlessly reached out since it depends on the object model.

b. Platform Free: contrary to other programming languages like C and C++, when Python is assembled, is not ordered into stage explicit machine, but instead into stage autonomous byte code. The byte code is conveyed via the web and decoded by the Virtual Machine on whichever stage it is being run on.

c. Simple: Python intends to be easy to learn on account of its English-like grammar and simple structure.

d. Secure: With Python's secure element it's able to develop a virus free, alter-free system. The validation method depends on open key encryption.

e. Architecture neutral: Python compiler draws out an architecture-neutral object file format, which makes the compiler code executable on numerous processors, with the presence of Python runtime system.

f. Portable: Being architecture-neural, having no execution subordinate parts of a particular makes Python versatile. The compiler in Python is written in ANSI C with a flawless versatility limit, which is a POSIX subset.

g. Robust: Python tries to avoid mistake inclined circumstances by batoning mainly on compile time error checking and runtime checking.

h. Multithreaded: Based on Python's multithreaded highlight, it is feasible to compose programs that can run many tasks at the same time. This design feature permits the developers to build intelligent applications that can operate as expected.

i. Interpreted: Python byte code is made an interpretation of on-the-fly to local machine guidelines and isn't stored anywhere. The advancement cycle is faster and more insightful since the connection is a gradual and lightweight process.

j. High Execution: With the use of Just-In-Time compilers, Python empowers superior execution.

k. Distributed: Python is implemented for the distributed environment of the web.

l. Dynamic: Python is viewed as more unique than C or C++ since it is designed to adjust to a developing climate. Python programs can convey a broad measure of run-time information that can be used to check and determine resolve accesses to objects on runtime.

The model however was developed over an Integrated Development Environment (IDE) which is the Anaconda. Anaconda Navigator on the hand is a desktop graphical user interface application that is distributed alongside the Anaconda Navigator. The major function is that it allows users to launch applications that are packaged in the Anaconda distribution with the use of command line interfaces. There are many tools ensuite in the Anaconda IDE, however, to develop the deep learning model, the system employs the use of a Jupyter Note pad. A Jupyter Notepad record is a browser-based REPL containing an arranged list of input/output cells which can contain code, messages (utilizing Markdown), math, plots and rich media. Under the connection point, a notebook is a JSON document, following a formed composition, normally finishing with the ".ipynb" expansion. Jupiter Scratchpad can associate with numerous kernels to permit programming in various dialects. A Jupyter kernel is a program liable for handling different kinds of packets (code execution, code fruitions, review), and answering. kernels converse with different parts of Jupyter using ZeroMQ and subsequently could be something very similar or remote machines. Dissimilar to many other notebook-like interfaces, in Jupyter, kernels don't know that they are attached to a particular record, and can be connected with numerous clients on the double. Typically kernels permit the execution of only a single language, yet there are several special cases Of course Jupyter Note pad ships with the IPython kernel. A Jupyter Journal can be changed over completely to varieties of open standard result designs (HTML, show slides, Plastic, PDF, ReStructuredText, Markdown, Python) through "Download As" on the internet, with the use of nbconvert library or "jupyter nbconvert" command line interface in a shell. To improve on the

representation of Jupyter notepad reports on the web, the nbconvert library is offered as a support via NbViewer] which can take a URL to any openly accessible notepad record, transform it to HTML spontaneously and show it to the client.

4.3 Result and Discussions

The system implemented two deep learning neural network models (BiLSTM and GRU). The figure 4.1 shows the performance of the BiLSTM model given the number of epochs and the Val accuracy per epochs. The system performs moderately well with maximum accuracy of 0.95 relative to 95% accuracy with the first dataset. While figure 4.2 shows the performance of the GRU model given the number of epochs and the accuracy of 0.92 relative to 92% accuracy with the first dataset. However, figure 4.3 and 4.4 shows the performances of BI-LSTM and GRU models given the number of epochs and accuracy of 0.61 and 0.65 respectively. The accuracy of the system increases as the number of epochs increases. There are a few points of collision between the training and validation caused by inconsistency in the dataset. However, the BI-LSTM Model performed much better with the IDS_ISCX_2012 dataset.

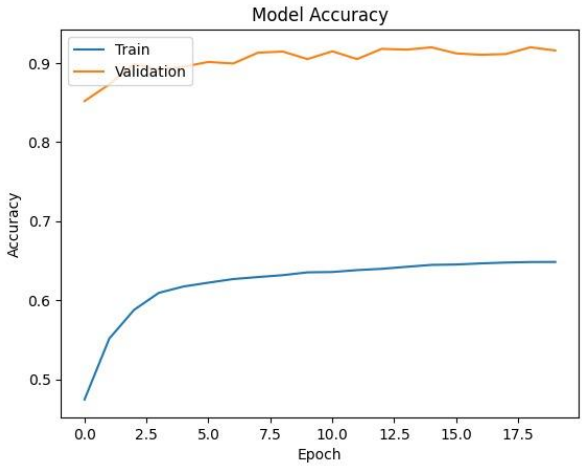

Figure 4.1 BI-LSTM

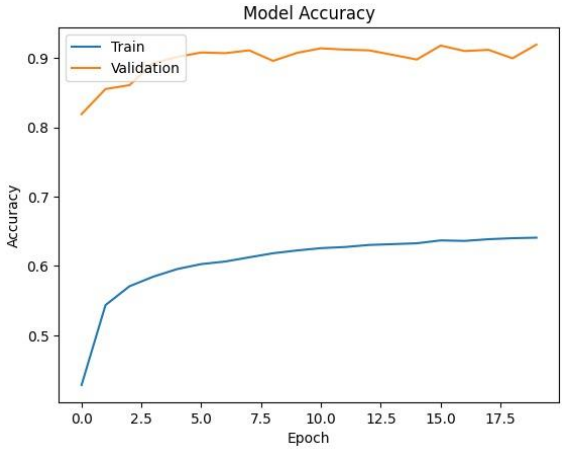

Figure 4.2 GRU

The graph 4.1 and 4.2 above are for CICDDOS2019 dataset.

Fig 4.3 BI-LSTM

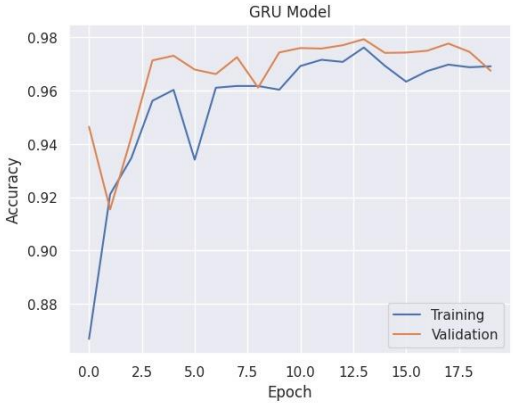

Fig 4.4 GRU

The graph 4.3 and 4.4 above are for IDS_ISCX_2012 dataset

In the context of evaluating neural networks, the loss to epochs graph is a visualization that illustrates the performance of a model during the training process. It tracks the changes in the loss function value over a series of training epochs. Each epoch represents a complete pass through the training dataset, where the model adjusts its internal parameters (weights and biases) to minimize

the loss and improve its predictive capability. The loss function measures the disparity between the predicted output of the neural network and the actual target output. It quantifies how well the model is capturing the underlying patterns and relationships in the training data. The goal of training is to iteratively update the model's parameters, reducing the loss and improving its ability to generalize to unseen data.

The loss to epochs graph typically displays the loss value on the y-axis and the number of epochs on the x-axis. At the beginning of training, the model's parameters are usually initialized randomly, resulting in high loss values. As the training progresses, the model gradually learns from the training examples and adjusts its parameters to better fit the data. Also, the loss to epochs graph aids in determining the optimal number of training epochs. Continuing training beyond a certain point can lead to over fitting, where the model becomes too specialized for the training data. On the other hand, stopping training too early may result in under fitting, where the model fails to capture the underlying patterns. By analysing the loss to epoch's graph, one can identify the point where validation loss starts to increase or reaches a plateau, indicating the optimal stopping point for training.

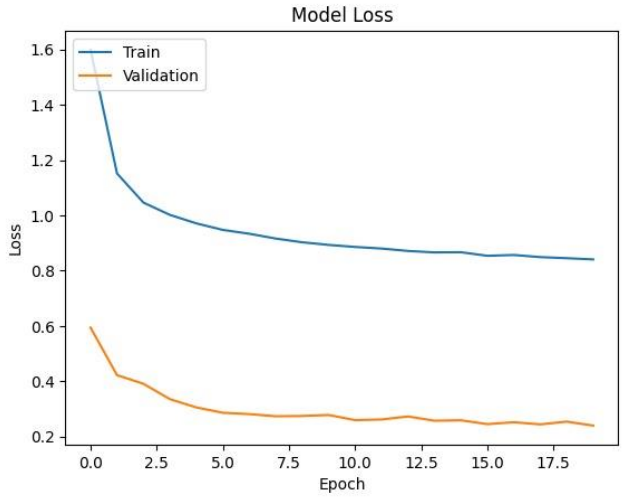

Figure 4.5 GRU

Figure 4.5 represent the graph of the Loss-Epoch relationship

Figure 4.5 shows that the model perform effectively and loss values are not decreasing, this shows a model well fitted and the plateau region is not consistent in the two graphs. Below is the table representing the evaluations of the two datasets and the models:

Table 4.1: Table representing the evaluations on the two datasets and the model.

Model	Dataset	F1	Accuracy	Specificity	Sensitivity	Precision
BI-LSTM	IDS_ISCX_2012	0.92	0.96	1.0	0.92	1.0
BI_LSTM	CICDDOS2019	0.98	0.91	0.89	0.91	0.65
GRU	IDS_ISCX_2012	0.81	0.92	0.72	0.94	0.72
GRU	CICDDOS2019	0.68	0.87	0.75	0.82	0.70

Table 4.1 shows the performance evaluation consisting of the F1-score, accuracy, specificity, sensitivity, and precision metrics for different models (BI-LSTM and GRU) on two datasets (IDS_ISCX_2012 and CICDDOS2019).

F1 Score: F1 Score = 2* (Precision * Recall) / (Precision + Recall)

Accuracy: Accuracy = (TP + TN) / (TP + TN + FP + FN)

Specificity: Specificity = TN / (TN + FP)

Sensitivity: Sensitivity = TP / (TP + FN)

Precision: Precision = TP / (TP + FP)

Recall = Sensitivity (same as sensitivity)

Where: TP = True Positive, TN = True Negative, FP = False Positive and FN = False negative.

For the BI-LSTM on IDS_ISCX_2012 dataset,

True Positives: 6058

True Negatives: 3457

False Positives: 9

False Negatives: 506 whereas, for GRU on the same dataset the following figures were obtained.

True Positives: 6856

True Negatives: 674

False Positives: 268

False Negatives: 405

CHAPTER FIVE

Conclusion and Recommendations

5.1 Conclusions

The research work targets the use of multiple datasets (IDS_ISCX_2012 and CICDDOS2019) for identifying DDOS attacks on cloud networks. The research work implements two (2) RNN techniques which are the Bidirectional Long Short Term Memory and Gated Recurrent Units.

The research disclosed the key features of the attacks as well as the advantages and disadvantages of different defense mechanisms. There is a tendency that the research does not cover all the known classes of attacks due to the limited scope. Although the dataset used in this research work was carefully selected to contain many classes of attack which enable this work to be a bedrock for a better model. This work also provides some discussions about DDoS attacks on other approaches that are possible in solving the problem statement

5.2 Recommendations and future work

The result obtained from this work has enabled me to recommend the following:

1. For effectiveness in detecting and mitigating the attacks caused by DDOS, the use of Deep Neural Networks is much more recommended as the DNNs have the capacity of selecting features dynamically.
2. This research work has shown that a single dataset is not sufficient to determine the effectiveness of deep learning models.
3. The use of ensemble techniques will foster better efficiency of the model development.

5.3 Limitations of the Study

The research work primarily focuses on the Distributed denial of service. The efficiency of the model is based on the type of datasets and the method of collection of the datasets. This project does not emphasize other types of attacks that are possible on a cloud-based network. However, the multiversity nature of the CICIDS Datasets can make it usable in another setup.

5.4 Contribution to Knowledge

This research work has made the following contributions:

a. A framework for detecting Distributed Denial of Service attack in a cloud-based environment using BI-LSTM and GRU Algorithm.
b. Using more than one datasets to see the performance of the model on multiple datasets which other researchers have not done.
c. Introduction of Temporal Attention Mechanism to focus on relevant patterns and capture crucial temporal dependencies.

5.5 Future Works

Future works will include the use of multiple models and the use of multi-layer ensemble. This will improve the precision and correctness of the model.

The use of an unsupervised learning approach will also be experimented with to encourage less dependency on data.

REFERENCE

Aborujilah, A., & Musa, S. (2017). Cloud-based detection of HTTP DDoS attack using hidden semi-Markov model with improved temporal fingerprints. Procedia Computer Science, 124, 85–93.

Albayram, Y., & Yayla, M. (2021). A systematic review of phishing and social engineering attacks: The human element in cybersecurity. Journal of Cybersecurity and Privacy, 1(3), 512–536.

Alharbi, A., & Khan, K. M. (2020). A survey of SQL injection attack detection and prevention. Journal of Computer Virology and Hacking Techniques, 16(3), 205–219.

Alzahrani, R. J., & Hong, L. (2023). Evolution and trends in DDoS attack mitigation: A systematic review. Computers & Security, 124, 102961.

Behal, S., Kumar, K., & Sachdeva, M. (2018). D-FACE: An anomaly based distributed approach for early detection of DDoS attacks and flash events. Journal of Network and Computer Applications, 111, 49–63.

Chen, Z., Wang, L., & Li, H. (2022). A bidirectional LSTM-based approach for network intrusion detection. IEEE Access, 10, 34567–34578.

Hameed, S., & Ali, U. (2018). HADEC: Hadoop-based live DDoS detection framework. EURASIP Journal on Information Security, 2018(1), 1–12.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.

Hoque, N., Bhuyan, M. H., Baishya, R. C., Bhattacharyya, D. K., & Kalita, J. K. (2017). Network attacks: Taxonomy, tools and systems. Journal of Network and Computer Applications, 40, 307–324.

Islam, M. S., Islam, M. R., & Ahamed, S. I. (2019). A survey on malware attacks and their detection. International Journal of Computer Applications, 178(41), 18–25.

Jiang, Y., Li, W., & Wang, J. (2022). Botnet-based DDoS attacks in cloud computing: Challenges and solutions. Future Generation Computer Systems, 135, 187–201.

Kumar, S., & Singh, K. (2021). Security and privacy issues in cloud computing: A comprehensive review. Journal of Cloud Computing, 10(1), 1–32.

Kumar, S. (2018). Long Short-Term Memory (LSTM) networks. In S. S. Roy (Ed.), Deep learning for natural language processing (pp. 45–62). Springer.

Lee, H., & Kim, J. (2023). A comparative study of GRU and LSTM for real-time intrusion detection. Journal of Information Security and Applications, 68, 103263.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444.

Li, F., Zhang, H., & Wang, X. (2021). Man-in-the-middle attacks in IoT networks: A survey. IEEE Internet of Things Journal, 8(12), 10235–10252.

Mell, P., & Grance, T. (2011). The NIST definition of cloud computing (Special Publication 800-145). National Institute of Standards and Technology.

Mishra, A., Pilli, E. S., & Joshi, R. C. (2021). Economic denial of sustainability (EDoS) in cloud computing: A survey. Computers & Security, 108, 102354.

Mousavi, S. H. (2014). Early detection of DDoS attacks in software defined networks [Doctoral dissertation, University of New Brunswick].

Patel, N., & Joshi, A. (2022). Mitigation strategies for multi-vector DDoS attacks in SDN-based cloud environments. International Journal of Network Management, 32(1), e2175.

Ristenpart, T., Tromer, E., Shacham, H., & Savage, S. (2020). Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds. ACM Transactions on Information and System Security, 13(2), 1–31.

Singh, K., & De, T. (2017). MLP-GA based algorithm for detection of DDoS attacks in cloud. International Journal of Computer Applications, 165(11), 30–34.

Singh, S., Sharma, P. K., Moon, S. Y., Moon, D., & Park, J. H. (2018). A comprehensive study on DDoS attacks and defense mechanisms in cloud computing. Journal of Supercomputing, 74(10), 5002–5037.

Sreeram, I., & Vuppala, V. P. K. (2017). HTTP flood attack detection in application layer using machine learning metrics and bio-inspired bat algorithm. Applied Computing and Informatics, 13(1), 69–80.

Wang, H., Zhang, D., & Zhao, Y. (2022). A deep learning-based DDoS detection system for cloud computing environments. Journal of Cloud Computing, 11(1), 1–15.

Wang, L., Cheng, H., Mohaisen, D., & Nguyen, P. (2014). Delving into internet DDoS attacks by botnets: Characterization and analysis. *IEEE/ACM Transactions on Networking, 22*(6), 2022–2035.

Zhang, J., Liu, Q., & Xu, M. (2023). A temporal attention-based GRU model for multi-vector DDoS attack detection in cloud networks. IEEE Transactions on Dependable and Secure Computing, 20(1), 123–135.

Zhang, Y., Li, X., & Wang, Z. (2022). Security challenges and solutions for cloud computing: A survey. Computers & Electrical Engineering, 99, 107835.