

*A Project report
on*

PREDICTION OF DDOS ATTACK USING DEEP LEARNING

*Submitted in partial fulfillment of the
requirements for the award of the degree
of*

BACHELOR OF TECHNOLOGY

in

Computer Science & Engineering

By

N. JYOTHI	(204G1A0546)
G. MAHESH KUMAR	(204G1A0551)
V. MEGHANA	(204G1A0556)
J. ASHOK	(204G1A0517)

Under the Guidance of

Dr. C. Sasikala, M.Tech, Ph.D.

Associate Professor



Department of Computer Science & Engineering

**SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY
(AUTONOMOUS)**

Rotarypuram Village, BK Samudram Mandal, Ananthapuramu-515701

2023-2024

**SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY
(AUTONOMOUS)**

**Affiliated to JNTUA, Accredited by NAAC with 'A' Grade, Approved by AICTE, New
Delhi & Accredited by NBA (EEE, ECE & CSE)**

Rotarypuram Village , B K Samudram Mandal , Ananthapuramu - 515701

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



Certificate

This is to certify that the Project Stage 1 report entitled **PREDICTION OF DDOS ATTACK USING DEEP LEARNING** is the bonafide work carried out by **N. Jyothi** bearing Roll Number **204G1A0546**, **G. Mahesh Kumar** bearing Roll Number **204G1A0551**, **V. Meghana** bearing Roll Number **204G1A0556**, **J. Ashok** bearing Roll Number **204G1A0517** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** during the academic year 2023-2024.

Guide

Dr. C. Sasikala, M.Tech, Ph.D.

Associate Professor

Head of the Department

Mr. P. Veera Prakash, M.Tech, (Ph.D.)

Assistant Professor & HOD

Date:

External Examiner

Place: Rotarypuram

DECLARATION

We, Ms. N. Jyothi bearing reg no: 204G1A0546, Mr. G. Mahesh Kumar bearing reg no: 204G1A0551, Ms. V. Meghana bearing reg no: 204G1A0556, Mr. J. Ashok bearing reg no: 204G1A0517, students of SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY, Rotarypuram, hereby declare that the dissertation entitled “PREDICTING DDOS ATTACK USING DEEP LEARNING” embodies the report of our project work carried out by us during IV Year Bachelor of Technology under the guidance of Dr. C. Sasikala, Associate Professor, Department of CSE and this work has been submitted for the partial fulfillment of the requirements for the award of Bachelor of Technology degree.

The results embodied in this project report have not been submitted to any other Universities of Institute for the award of Degree.

N. JYOTHI

Reg no: 204G1A0546

G. MAHESH KUMAR

Reg no: 204G1A0551

V. MEGHANA

Reg no: 204G1A0556

J. ASHOK

Reg no: 204G1A0517

VISION AND MISSION OF THE INSTITUTION

Vision:

To become a premier Educational Institution in India offering the best teaching and learning environment for our students that will enable them to become complete individuals with professional competency, human touch, ethical values, service motto, and a strong sense of responsibility towards environment and society at large.

Mission:

- Continually enhance the quality of physical infrastructure and human resources to evolve in to a center of excellence in engineering education.
- Provide comprehensive learning experiences that are conducive for the students to acquire professional competences, ethical values, life-long learning abilities and understanding of the technology, environment and society.
- Strengthen industry institute interactions to enable the students work on realistic problems and acquire the ability to face the ever changing requirements of the industry.
- Continually enhance the quality of the relationship between students and faculty which is a key to the development of an exciting and rewarding learning environment in the college.

VISION AND MISSION OF THE DEPARTMENT OF CSE

Vision:

To evolve as a leading department by offering best comprehensive teaching and learning practices for students to be self-competent technocrats with professional ethics and social responsibilities.

Mission:

DM 1: Continuous enhancement of the teaching-learning practices to gain profound knowledge in theoretical & practical aspects of computer science applications.

DM 2: Administer training on emerging technologies and motivate the students to inculcate self-learning abilities, ethical values and social consciousness to become competent professionals.

DM 3: Perpetual elevation of Industry-Institute interactions to facilitate the students to work on real-time problems to serve the needs of the society.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that we have now the opportunity to express our gratitude for all of them.

It is with immense pleasure that we would like to express our indebted gratitude to our Guide **Dr. C. Sasikala, Associate Professor, Computer Science & Engineering Department**, who has guided us a lot and encouraged us in every step of the project work. We thank her for the stimulating guidance, constant encouragement and constructive criticism which have made possible to bring out this project work.

We express our deep felt gratitude to **Mr. C. Lakshminatha Reddy, Assistant Professor and Mr. M. Narasimhulu, Assistant Professor, Project Coordinators** for their valuable guidance and unstinting encouragement enabled us to accomplish our project successfully in time.

We are very much thankful to **Mr. P. Veera Prakash, Assistant Professor & Head Of the Department, Computer Science & Engineering**, for his kind support and for providing necessary facilities to carry out the work.

We wish to convey our special thanks to **Dr. G. Balakrishna, Principal of Srinivasa Ramanujan Institute of Technology** for giving the required information in doing our project work. Not to forget, we thank all other faculty and non-teaching staff, and our friends who had directly or indirectly helped and supported us in completing our project in time.

We also express our sincere thanks to the Management for providing excellent facilities.

Finally, we wish to convey our gratitude to our family who fostered all the requirements and facilities that we need.

PROJECT ASSOCIATES

204G1A0546

204G1A0551

204G1A0556

204G1A0517

ABSTRACT

In recent years, Internet services have been increased in public and business ventures for production tasks. So internet applications need a lot of security to secure the data of other businesses and also itself. DDoS attacks are major security risks in the application environment. It happens by sending thousands of requests to flood the server and prevent it from processing requests. DDoS attack is a significant cybersecurity challenge that makes a particular system or network out of reach and unusable for some time. It affects the server's resources. The proposed system is used to detect such types of attacks by utilizing LSTM algorithm and a high level of accuracy. Hence, this work aims to solve this issue by applying a Long Short-Term Memory(LSTM) algorithm with a high degree of accuracy to detect these types of assaults. The suggested technique, which has a 93% accuracy rate in identifying DDoS attacks, will be evaluated and simulated using Python and it is compared with the existing machine learning algorithms.

Keywords:

Deep Learning, Long-short term memory, Distributed Denial-of-service(DDoS)attacks

CONTENTS		Page No.
List of Figures		IX
List of Tables		X
List of Abbreviations		XI
Chapter 1	Introduction	1-4
	1.1 Real-time of attacker disrupting the user	3
	1.2 Deep Learning	4
	1.3 Objective	4
Chapter 2	Literature Survey	5-6
Chapter 3	System Design	7-11
	3.1 System model of DDoS attack	7
	3.2 Work Flow of Proposed System	9
	3.3 Dataset Preprocessing	10
	3.3.1 Data Cleaning	10
	3.4 Deep Learning	10
Chapter 4	Methodology and Algorithms	12-18
	4.1 Module	12
	4.1.1 User	13
	4.1.2 System	13
	4.2 Gradient Boosting, Decision Tree and LSTM	14
	4.2.1 Gradient Boosting	14
	4.2.2 Decision Tree	14
	4.2.3 RNN	15
	4.2.4 Working of LSTM	16
Chapter 5	System Requirement Specification	19-23
	5.1 Python	20
	5.2 Visual Studio code	20
	5.3 Modules Used	20
	5.3.1 Matplotlib	21
	5.3.2 SkLearn	21
	5.3.3 Seaborn	22
	5.3.4 Pandas	22
	5.3.5 NumPy	22
	5.4 Time	23
Chapter 6	Implementation	24-29
	6.1 Dataset	24
	6.2 Source Code	24
	6.2.1 Data Collection	24

	6.2.2 Data Preprocessing	26
	6.2.2.1 Setting dataset dimensions	26
	6.2.2.2 Concise Summary of dataset	26
	6.2.3 Model Creation and Model Training	27
	6.2.4 Model Testing	28
	6.2.4.1 Testing with LSTM	28
	6.2.4.2 Testing with Decision Tree	29
	6.2.4.3 Testing with Gradient Boosting	29
Chapter 7	Result Analysis	30-35
	7.1 Data pre-processing	30
	7.2 Label Encoder	31
	7.3 Evaluation Metrics	31
	7.4 Website	33
	Conclusion	36
	References	37-38
	Publication paper	
	Participation Certificate	

LIST OF FIGURES

Fig. No	Figure Name	Page No.
1.1	DDoS attack using Botnet	11
1.2	DDoS Attacker disrupting the user	12
3.1	Architecture of DDOS attacks	16
3.2	Work Flow of the Proposed System	18
3.3	Data Preprocessing	19
4.1	GUI Interface	21
4.2	Use case diagram	22
4.3	Simple RNN	24
4.4	Feed- Forward Neural Network	24
4.5	LSTM cell with activation functions	25
4.6	LSTM cell	26
6.1	Dataset Used	33
6.2	Libraries imported	34
6.3	Size of dataset	35
6.4	Concise summary of dataset	35
6.5	LSTM model creation	36
6.6	Output of model creation	37
6.7	Accuracy of LSTM	37
6.8	Accuracy of Decision Tree	38
6.9	Accuracy of Gradient Boosting	38
7.1	Heat-map of missing values	39
7.2	Traffic for normal and Malicious traffic	39
7.3	Comparison of LSTM and GB	41
7.4	Comparison of LSTM and DT	41
7.5	Comparison Graph	42
7.6	Home page	42
7.7	Load page	43
7.8	View Data	43
7.9	Select Model	44
7.10	Prediction page	44

LIST OF TABLES

Table.No	Title	Page No.
5.1	Project stages	38
7.1	Tabular Representation of Models	45

LIST OF ABBREVIATIONS

ANN	Artificial neural networks
DFD	Data flow diagram
OO	Object oriented
UML	Unified Modeling Language
DL	Deep learning
ML	Machine Learning
DNN	Deep Neural Networks
RNN	Recurrent Neural Networks
DOS	Denial of Service
DDoS	Distributed Denial of Service
TCP	Transition Control Protocol
UDP	User Datagram Protocol
ICMP	Internet Control Message Protocol
SDN	Standard Normal Deviation
IoT	Internet Of Things

CHAPTER 1

INTRODUCTION

A distributed denial-of-service (DDoS) attack is a deliberate attempt to disrupt normal operations of a server, service, or network by overloading the target or the infrastructure around it with too much traffic. The effectiveness of DDoS assaults originates from their capacity to leverage a large number of compromised computer systems as attack traffic sources. Machines that are networked and have Internet of Things devices could be deemed as exploited machines. At a high level, a denial-of-service attack (DDoS) might be likened to an unexpected traffic jam that closes a highway and prevents regular traffic from getting to its intended destination. DDoS assaults make use of computer networks that are online.

These networks are made up of computers and other devices (such Internet of Things devices) that have been infected with malware, enabling an attacker to remotely manipulate them. These standalone devices are known as bots (sometimes called zombies), and a collection of bots is known as a botnet. An attacker can control an attack by remotely instructing each bot in the botnet once it has been set up as shown in Figure 1. Every bot that is sent to an IP address that is the target of a botnet attack sends queries to the IP address of the victim, which may overwhelm the server or network and cause a denial of service to regular traffic. It might be challenging to distinguish between attack and legal traffic because every bot is an Internet device.

The most noticeable sign of a denial-of-service assault is when a website or service suddenly becomes unreliable or slow. However, since several factors, such a real traffic increase, can result in comparable performance problems, more research is typically necessary. You can detect some of these obvious indicators of a DDoS assault with the aid of traffic analytics tools. Unusual volumes of traffic coming from a single IP address or range, a deluge of traffic from users with similar device types, geolocation settings, or web browser versions, or an inexplicable spike in requests to a single page or endpoint are all signs of suspicious activity. Unusual traffic patterns, such spikes at strange times of day or patterns that seem out of the ordinary (like a spike every 10 minutes).

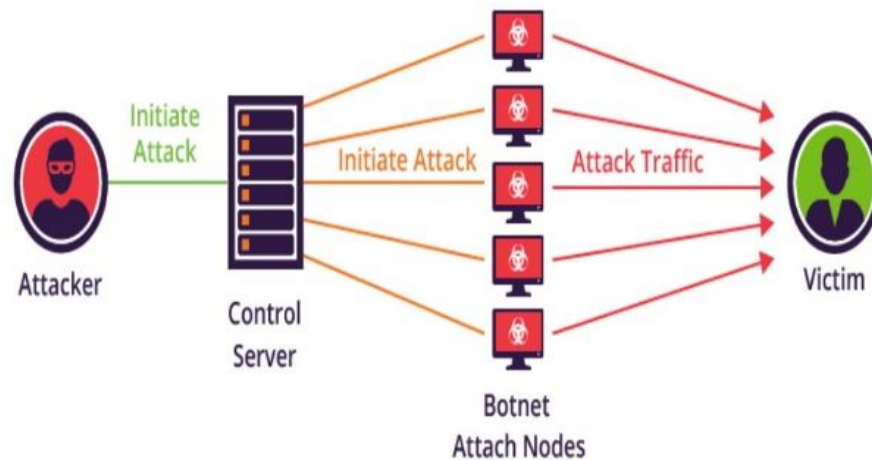


Figure 1.1 : DDoS Attack using Botnet[13]

Some of the famous DDoS attacks on some organizations such as, on 28 February 2018 The largest-ever DDoS attack was launched against GitHub, a well-known online code management site utilized by millions of developers. platform was not ready for the enormous influx of traffic, which peaked at a record-breaking 1.3 terabits per second, even though it was accustomed to high levels of traffic. The GitHub attack used a technique called memcaching, a database caching solution meant to speed up networks and websites, rather than botnets. After successfully impersonating GitHub, the attackers significantly increased the volume of traffic going to the platform. Thanks to the DDoS protection solution that GitHub was utilizing, the attack was contained and prevented from spreading in less than ten minutes after it started.[14]

October 2016 saw the second-largest DDoS attack against major DNS operator Dyn. The hack caused significant disruption, bringing down the websites of over 80 of its clients, including Reddit, Amazon, Netflix, Airbnb, Spotify, Twitter, and PayPal. Hackers built a vast botnet of 100,000 Internet of things (IoT) devices to launch their attack using a malware known as Mirai. Radios, smart TVs, and printers were among the gadgets that were set up to bombard Dyn with requests and cause traffic congestion. Approximately 14,500 domains stopped using Dyn's services immediately after the attack, which is estimated to have caused \$110 million in damage even though it was contained in a single day.[14]

Ransomware and DDoS assaults were identified as the top two threats affecting businesses in 2018 by the UK's National Crime Agency. They saw a sharp rise in attacks and recommended that organizations take urgent action to fortify themselves against this escalating danger.

This lengthy list makes it clear that DDoS assaults have the power to bring down entire corporate websites, networks, and, as the Dyn attack showed, nearly the whole internet.

Businesses ought to think about utilizing a DDoS protection service, which can identify unusual traffic patterns and divert DDoS attacks off the network. Additional security precautions include using firewalls, VPNs, anti-spam software, and additional DDoS defense layers to safeguard network infrastructure.

1.1.Real-time of attacker disrupting user

DDoS attacks involve malicious efforts to overwhelm a target system or network with an excessive volume of traffic, causing disruption and rendering the services inaccessible to legitimate users. The dynamic and distributed nature of cloud infrastructures further complicates the detection and mitigation of such attacks. Traditional security measures, while effective to some extent, are often insufficient in addressing the evolving sophistication of DDoS attacks.

In recent years, the rapid proliferation of cloud computing has revolutionized the way organizations manage and deploy their IT infrastructure. Cloud environments offer scalability, flexibility, and cost efficiency, making them an attractive choice for hosting critical applications and services. However, this widespread adoption has also exposed cloud systems to an escalating threat landscape, with Distributed Denial of Service (DDoS) attacks emerging as a formidable challenge.

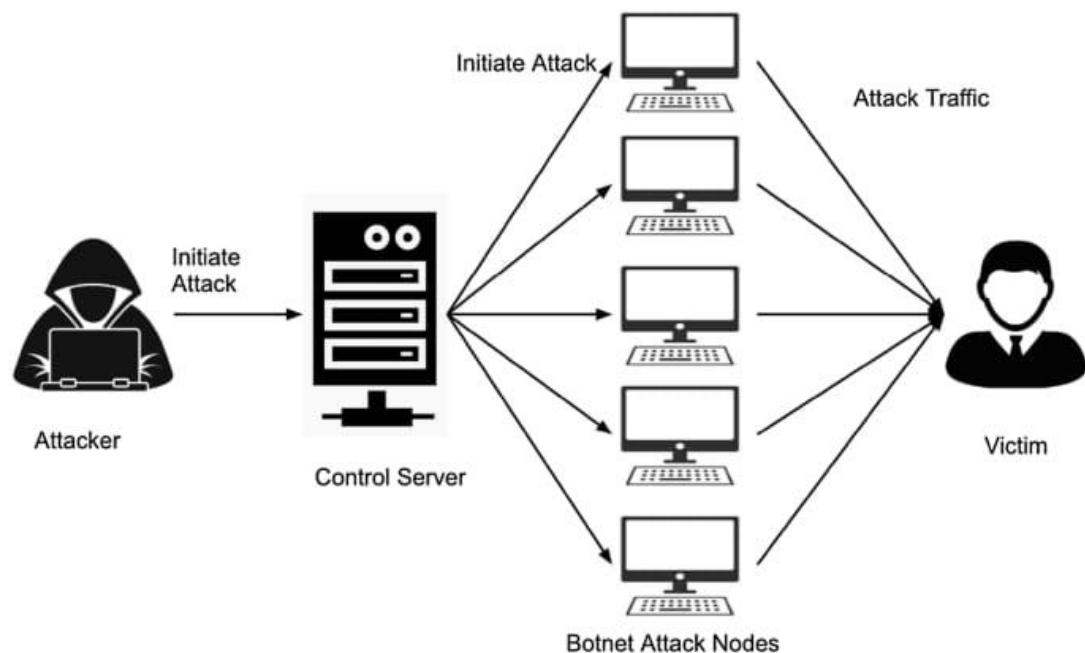


Figure 1.2: DDoS attacker disrupting the user

The figure 2 explains how the attacker attacks the network using a botnet and disrupts the usage of the normal user to access the server. By using botnet attackers increase the traffic over the internet and server, which makes the user unable to reach the server.

1.2.Deep Learning

This work focus on leverage the power of deep learning techniques for the prediction and early detection of DDoS attacks in cloud environments. Deep learning, a subset of machine learning, has demonstrated remarkable capabilities in extracting intricate patterns and features from complex data sets. By harnessing the inherent adaptability of deep learning algorithms, this project seeks to enhance the ability to identify and respond to DDoS threats in real time, thereby fortifying the security posture of cloud-based systems.

1.3 Objective

The objectives of this study encompass the development of a robust deep-learning model trained on historical data to recognize subtle patterns indicative of impending DDoS attacks. Additionally, the project will explore the integration of anomaly detection mechanisms to augment the model's ability to discern abnormal network behavior. The ultimate goal is to create an intelligent and proactive defense system capable of predicting and mitigating DDoS attacks before they can inflict significant damage. The objectives of the paper is three methodologies:

- i. To develop a Deep Learning model (LSTM) to detect the attack.
- ii. To compare the model with the Machine Learning models.
- iii. To create a model which can be used for real-time detection of DDos attacks.

Through this work, aspire to contribute to the advancement of security by providing a predictive framework that empowers organizations to safeguard their critical assets and ensure uninterrupted service delivery in the face of evolving cyber threats. The outcomes of this project hold the potential to redefine the landscape of DDoS defense in the cloud, fostering a more resilient and secure digital environment for businesses and individuals alike.

CHAPTER 2

LITERATURE SURVEY

It is unimaginable that a single attack could result in so significant damage to a computer system or network. However, due to its nature, DDoS will actually bring down the entire network. Its prevention is consequently very difficult to achieve. As a result, there is a huge demand for effective frameworks for DDoS attack detection. Several writers have developed several approaches to identify DDoS attacks in response to this demand. A few of them are detailed it has advantages and disadvantages:

Ankit Agarwal.[1] Most of the methods cannot simultaneously achieve efficient detection with a small number of false alarms. In this case, deep learning techniques are appropriate and effective algorithms to categorize both normal and attacked information. Hence, a novel feature selection-whale optimization algorithm deep neural network (FS-WOA-DNN) method is proposed in this research article to mitigate DDoS attack effectively. Initially, a pre-processing step is carried out for the input dataset where a min-max normalization technique is applied to replace all the input in a specified range. Later on, that normalized information is fed into the proposed FSWOA to select the optimal set of features for ease of the classification process. Those selected features are subjected to a deep neural network classifier to categorize normal and attacked data.

Mohammad Shurman.[2] In this paper, they proposed two methodologies to detect Distributed Reflection Denial of Service (DDoS) attacks in IoT. The first methodology uses a hybrid Intrusion Detection System (IDS) to detect IoT-DoS attacks. The second methodology uses deep learning models, based on Long Short-Term Memory (LSTM) trained with the latest dataset for such kinds of DDoS

Chen Zhibin[3] In this work, they apply a Hybrid Deep Learning method to detect malicious web traffic in the form of DDoS attacks, controlling the web flow of information reaching a server, and using any dependencies between the different elements of a data stream. An original and cutting-edge Hierarchical Temporal Memory (HTM) hybrid model has been proposed. (e operation of this model is predicated primarily on the portion of the cerebral cortex known as the neocortex. (The neocortex is in charge of various fundamental brain functions, including the perception of senses, the comprehension of language, and the control of movement.

Dong, S., & Sarem, M [4] The Distributed Denial of Service (DDoS) attack has seriously impaired network availability for decades and still there is no effective defense

mechanism against it. However, the emerging Software Defined Networking (SDN) provides a new way to reconsider the defense against DDoS attacks. In this paper, we propose two methods to detect the DDoS attack in SDN. One method adopts the degree of DDoS attack to identify the DDoS attack. The other method uses the improved K-Nearest Neighbors (KNN) algorithm based on Machine Learning (ML) to discover the DDoS attack. The results of the theoretical analysis and the experimental results on datasets show that our proposed methods can better detect the DDoS attack compared with other methods.

Abbas, K., & Jain, R[5] Recently, software defined networks (SDNs) and cloud computing have been widely adopted by researchers and industry. However, widespread acceptance of these novel networking paradigms has been hampered by the security threats. Advances in the processing technologies have helped attackers in increasing the attacks too, for instance, the development of Denial of Service (DoS) attacks to distributed DoS (DDoS) attacks which are seldom identified by conventional firewalls. In this paper, we present the state of art of the DDoS attacks in SDN and cloud computing scenarios. Especially, we focus on the analysis of SDN and cloud computing architecture. Besides, we also overview the research works and open problems in identifying and tackling the DDoS attacks.

Wang, Y [6], it is necessary to propose an effective method to detect DDoS attack from massive data traffics. However, the existing schemes have some limitations, including that supervised learning methods, need large numbers of labeled data and unsupervised learning algorithms have relatively low detection rate and high false positive rate. In order to tackle these issues, this paper presents a semi-supervised weighted k-means detection method. Specifically, in this paper, firstly present a Hadoop-based hybrid feature selection algorithm to find the most effective feature sets and propose an improved density-based initial cluster centers selection algorithm to solve the problem of outliers and local optimal. Then, we provide the Semi-supervised K-means algorithm using hybrid feature selection (SKM-HFS) to detect attacks. Finally, we exploit DARPA DDoS dataset, CAIDA “DDoS attack 2007” dataset, CICIDS “DDoS attack 2017” dataset and real-world dataset to carry out the verification experiment. The experiment results have demonstrated that the proposed method outperforms the benchmark in respect of detection performance and technique for order preference by similarity to an ideal solution (TOPSIS) evaluation factor.

As per the above papers, the conclusion is by DDoS attack the functioning of the system gets interrupted and the website stops working for the user. They developed several models but the accuracy was not up to the mark and it is less. So there is a need to improve the accuracy.

CHAPTER 3

SYSTEM DESIGN

The proposed model is an innovative application that can be considered a highly useful system, as it addresses the limitations commonly encountered with traditional and other existing methods for DDoS attack detection. The primary objective of this study is to develop a fast and reliable method that accurately detects the effects of Distributed Denial of Service (DDoS) attacks. In designing this system, we have leveraged the capabilities of a powerful algorithm in a Python-based environment, which includes the integration of Long Short-Term Memory (LSTM) neural networks.

3.1 System model of DDoS attack

A Distributed Denial of Service (DDoS) attack is a malicious attempt to disrupt the availability of a website, server, or network by overwhelming it with a flood of incoming traffic from multiple sources.

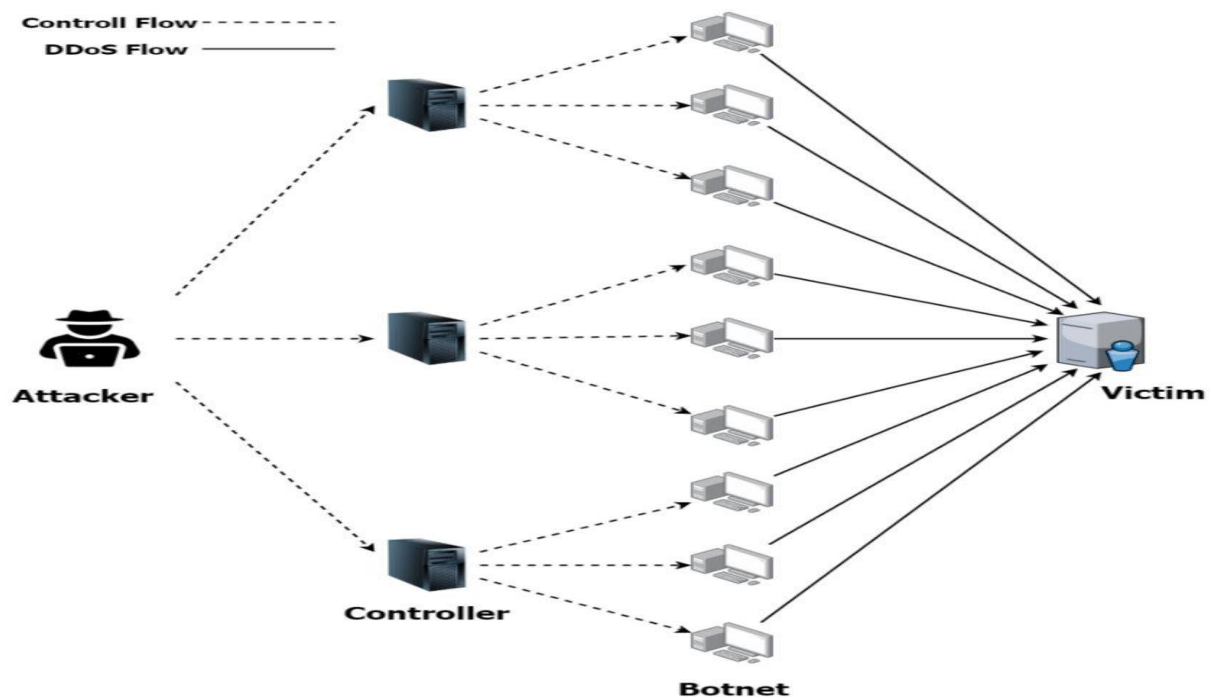


Figure 3.1 : A Architecture of Distributed Denial of Service (DDoS) attack

The architecture of a DDoS attack typically involves several key components:

Botnet: The attacker usually controls a large number of compromised computers, servers, or Internet of Things (IoT) devices that are collectively known as a botnet. These devices are often infected with malware that allows the attacker to control them remotely without the owners' knowledge. The botnet acts as a network of "bots" that can be instructed to generate and send massive amounts of traffic towards the target system.

Command and Control (C&C) Server: The attacker uses a central server or a group of servers, known as the Command and Control (C&C) server, to send instructions to the botnet. These instructions can include the target IP address or domain name, the type of attack to be launched, and the duration of the attack.

Attack Tools: The attacker uses specialized software or tools to automate and orchestrate the DDoS attack. These tools can include programs or scripts that control the botnet, generate malicious traffic, and launch various types of DDoS attacks such as volumetric attacks, protocol attacks, or application layer attacks.

Victim System: The target of the DDoS attack is the victim system, which can be a website, server, or network. The victim system is overwhelmed with an excessive amount of traffic from the botnet, causing it to become unresponsive or unavailable to legitimate users.

Spoofed IP Addresses: To make it difficult to trace the attack back to the original source, the attacker often spoofs or falsifies the IP addresses of the botnet devices. This makes it challenging for the victim system to block the attack based on the source IP addresses, as they appear to be coming from different locations.

Amplification Techniques: In some cases, the attacker may use amplification techniques to magnify the volume of traffic being sent to the victim system. For example, the attacker may use reflective amplification, where they send requests with a spoofed source IP address to vulnerable servers that respond with a much larger response to the victim system, overwhelming its resources.

Coordinated Timing: The attacker may coordinate the timing of the DDoS attack to maximize its impact. For example, launching the attack during peak hours of website traffic or during critical events to cause the most disruption.

In summary, the architecture of a DDoS attack involves a botnet controlled by a C&C server, attack tools to automate and orchestrate the attack, spoofed IP addresses to hide the attacker's identity, amplification techniques to magnify the attack, and coordinated timing for maximum impact. It's important to note that DDoS attacks are illegal and can cause significant harm to the targeted systems and organizations.

3.2 Work Flow of Proposed system

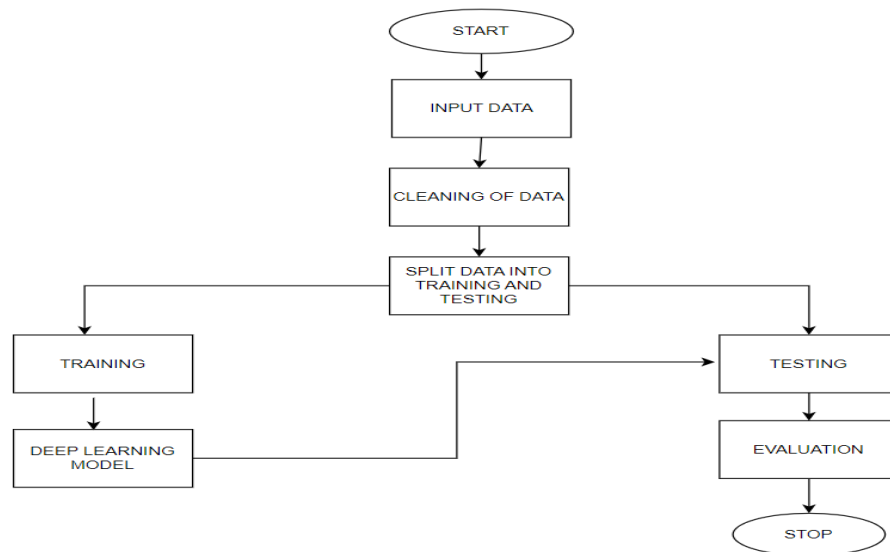


Figure 3.2 Work Flow of the Proposed System

Figure 3.2 shows how the proposed system is designed from starting such as taking dataset and preprocessing the data and splitting data into training and testing dataset. Deep Learning model is built using training dataset and trained it. Using testing dataset system is tested and evaluated the output.

3.2 Dataset Preprocessing

Preparing unprocessed data so that a deep learning model may use it is known as data preparation. To put it another way, when data is acquired in raw format from multiple sources, it is impractical for analysis. It is an essential first step in building a deep learning model. It is not always the case that we find clean, prepared data when developing a deep learning project. Additionally, data must always be cleaned and formatted before being used in any kind of activity. We therefore employ the data-preprocessing job for this. The procedures we must take to change data so that a machine can understand it are referred to as data preprocessing. The main agenda for a model to be accurate and precise in predictions is that the algorithm should be able to easily interpret the data's features.



Figure 3.3: Data Preprocessing

3.3 Data Cleaning

Data Cleaning is particularly done as part of data preprocessing to clean the data by filling missing values, smoothing the noisy data, resolving the inconsistency, and removing outliers.

- **Missing values**

Here are a few ways to solve this issue:

- **Ignore those tuples**

This method should be considered when the dataset is huge and numerous missing values are present within a tuple.

- **Fill in the missing values**

There are many methods to achieve this, such as filling in the values manually, predicting the missing values using regression method, or numerical methods like attribute mean

3.4 Deep Learning

Utilizing an LSTM (Long Short-Term Memory) model in our deep learning project, "Prediction of DDoS Attacks," is highly suitable due to its inherent capabilities in handling sequential data and capturing temporal dependencies. DDoS attacks often exhibit intricate patterns over time, making them akin to time-series data. LSTM, a type of recurrent neural

network (RNN), excels in processing such sequential information by maintaining long-term dependencies while mitigating the vanishing gradient problem encountered in traditional RNNs. Its gated architecture enables the model to selectively retain or forget information from previous time steps, allowing for the preservation of relevant features crucial for identifying DDoS attack patterns. Furthermore, LSTM's ability to learn from historical sequences enables it to adapt dynamically to evolving attack strategies, thereby enhancing the robustness of our prediction system. By leveraging the unique strengths of LSTM, we aim to develop a predictive model that can effectively discern DDoS attacks from normal network traffic, thus bolstering network security and preemptively mitigating potential threats.

CHAPTER 4

METHODOLOGY AND ALGORITHMS

4.1 Module:

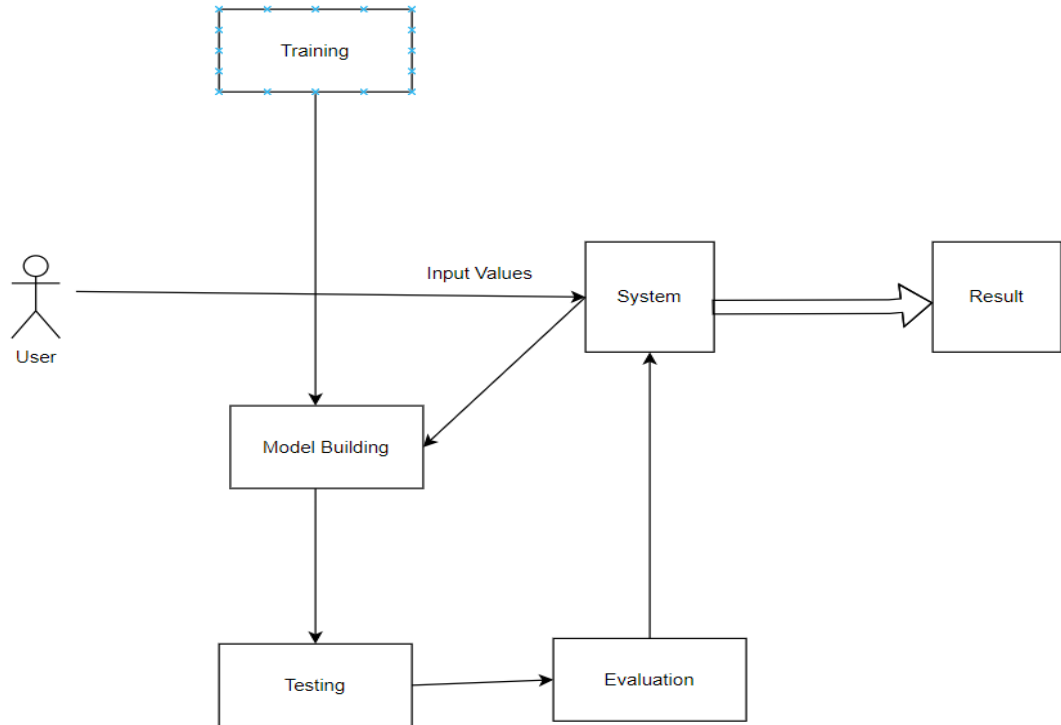


Figure 4.1 : GUI INTERFACE

The Figure 4.1 gives an overview of Graphical user Interface for the approach of the user to the system through the developed website.

4.1.1 User

Upload Dataset: In this module, users have the capability to upload their dataset, typically in a specified format (e.g., CSV, Excel, or database connection). The system should provide clear instructions on the accepted data format and structure, ensuring a seamless data upload process.

View Dataset: Users can view the dataset they have uploaded. The system may provide features for data visualization, filtering, and summary statistics to help users understand and explore the dataset before initiating the prediction process.

Input Values for Prediction: Users need to provide input values relevant to the prediction task. These inputs could include specific data points or variables necessary for the model to make predictions. The system should guide users on what input is required and validate the inputs to ensure they meet the necessary criteria.



Figure 4.2 : Usecase diagram

The Figure 4.2 gives usecase diagram of the activities done by the user and the system.

4.1.2 System

Take the Dataset : The system takes the dataset uploaded by the user and stores it securely. It performs data integrity checks and ensures that the dataset is available for further processing.

Preprocessing : In the preprocessing phase, the system cleans and prepares the data for model building. This involves handling missing values, data transformation, normalization, and feature engineering. It is a critical step to ensure the dataset is ready for training.

Training : The system utilizes machine learning or deep learning techniques to build a predictive model based on the preprocessed dataset. This may involve splitting the data into training and testing sets, selecting an appropriate algorithm, and training the model on the training data. The model is then evaluated for its performance on the testing data.

Generate Results: Once the model is trained, the system uses it to generate results. For a DDoS attack prediction system, this could mean evaluating whether the input values provided by the user are indicative of an attack or not. Results may be presented to the user in a user-friendly format, such as a binary classification (e.g., "Attack Detected" or "No Attack Detected") or with probability scores. Users may also receive insights or visualizations that help them understand the model's decisions.

4.2 Gradient Boosting, Decision Tree, LSTM Algorithms

4.2.1 Gradient Boosting

The gradient boosting algorithm is one of the most powerful algorithms in the field of machine learning. As we know the errors in machine learning algorithms are broadly classified into two categories i.e. Bias Error and Variance Error. As gradient boosting is one of the boosting algorithms it is used to minimize bias error of the model.

Unlike, the Adaboosting algorithm, the base estimator in the gradient boosting algorithm cannot be mentioned by us. The base estimator for the Gradient Boost algorithm is fixed i.e. Decision Stump. Like, AdaBoost, we can tune the `n_estimator` of the gradient boosting algorithm. However, if we do not mention the value of `n_estimator`, the default value of `n_estimator` for this algorithm is 100.

Gradient boosting algorithm can be used for predicting not only continuous target variable (as a Regressor) but also categorical target variable (as a Classifier). When it is used as a regressor, the cost function is Mean Square Error (MSE) and when it is used as a classifier then the cost function is Log loss.

4.2.2 Decision Tree

A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. Though a commonly used tool in data mining for deriving a strategy to reach a particular goal.

A decision tree is drawn upside down with its root at the top. In the image on the left, the bold text in black represents a condition/internal node, based on which the tree splits into branches/ edges. The end of the branch that doesn't split anymore is the decision/leaf, in this case, whether the passenger died or survived, represented as red and green text respectively.

Although, a real dataset will have a lot more features and this will just be a branch in a much bigger tree, but you can't ignore the simplicity of this algorithm. The feature importance is clear and relations can be viewed easily. This methodology is more commonly known as learning decision tree from data and above tree is called Classification tree as the target is to classify passenger as survived or died. Regression trees are represented in the same manner, just they predict continuous values like price of a house. In general, Decision Tree algorithms are referred to as CART or Classification and Regression Trees.

4.2.3 RNN

NEURAL NETWORK

A Neural Network consists of different layers connected, working on the structure and function of a human brain. It learns from huge volumes of data and uses complex algorithms to train a neural net. Here is an example of how neural networks can identify a dog's breed based on their features. The image pixels of two different breeds of dogs are fed to the input layer of the neural network. The image pixels are then processed in the hidden layers for feature extraction. The output layer produces the result to identify if it's a German Shepherd or a Labrador. Such networks do not require memorizing the past output. Several neural networks can help solve different business problems. Let's look at a few of them.

The functioning of a Recurrent Neural Network is based on the concept of retaining the output from a specific layer and reintroducing it as input, enabling the model to predict subsequent layer outputs, as illustrated in Figure 6.

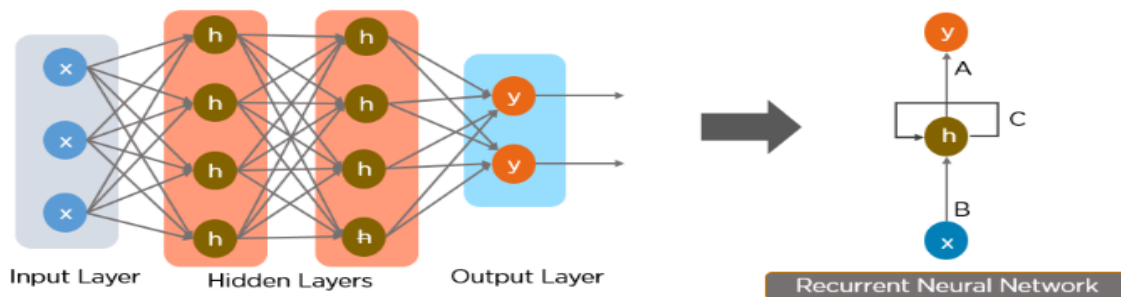


Figure 4.3: Simple Recurrent Neural Network

Feed-Forward Neural Networks

Figure 7 shows a simplified representation of a feed-forward-neural-network. A feed-forward NN restricts information flow to only one direction: forward from the input to output nodes via the hidden layers. The network does not contain any cycles or loops.

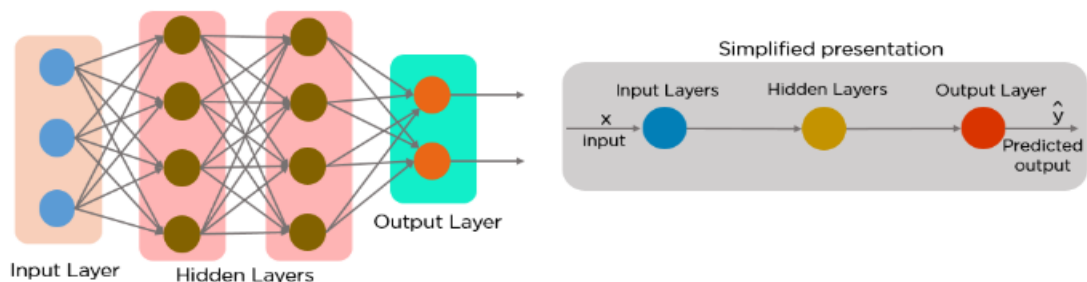


Figure 4.4 : Feed-forward NN

The inception of RNNs stemmed from challenges encountered in feed-forward neural networks, specifically their limitations in handling sequential data. Unlike feed-forward networks that focus solely on the current input without retaining information from previous inputs, RNNs address this issue by offering the capability to manage consecutive data and recall past inputs. Both the current inputs and previously received input can be handled in sequentially by an RNN. Because RNNs have internal memory, they can retain earlier inputs.

4.2.4 Working of LSTM cell

The development of recurrent neural networks stemmed from several challenges encountered in feed-forward neural networks. Unable to handle sequential data, it only considers the present input and lacks the ability to retain previous inputs. For these problems, the Recurrent-Neural Network (RNN) provides the answer. Both the current input and previously received input can be handled sequentially with RNN, cause RNNs have internal memory, they can retain earlier input. Recurrent Neural Networks (RNNs) share similarities with traditional neural networks but excel in capturing long-term dependencies, especially in task involving the sequence prediction. Unlike neural networks that focus on individual data points, LSTM stands out for its capacity to understand entire sequences due to the incorporation of feedback connections.

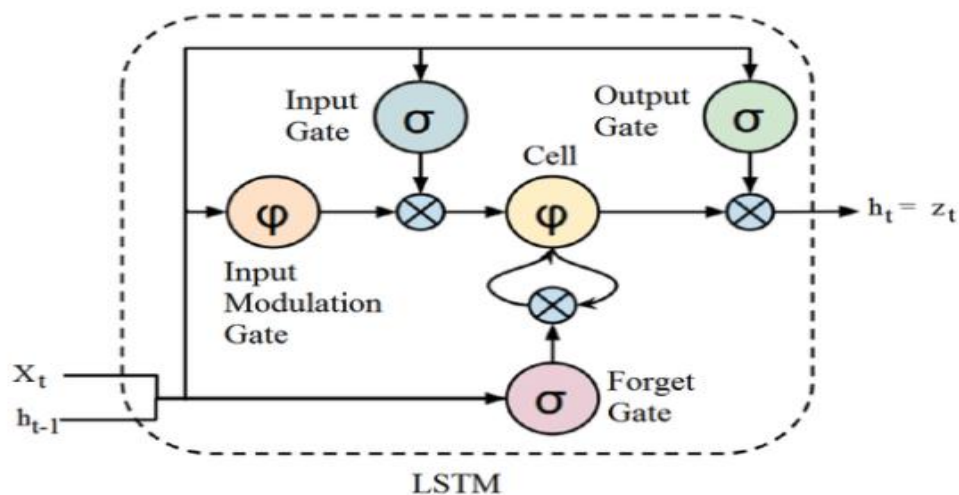


Figure 4.5 : LSTM cell

- A memory cell that sustains its state across time, referred to as a "cell state," plays a pivotal role in LSTM model. The horizontal line which that pass through the above or top of cell in LSTM cell as\ in Figure 8 represents the cell state. It might be seen as an information conveyor belt that information just moves across, unaltered.

- In LSTM, gates control the addition and deletion of data from the cell state. Information can optionally enter and exit the cell through these gates. To facilitate its operation, the system incorporates a sigmoid neural layer in conjunction with pointwise multiplication operation.
- Typically, the remember vector can also be refer as the forget gate. The forget gate's output multiplies 0 to a matrix point to notify the cell state what data to ignore. Information is retained if output of forget gate is 1, describe the status of the cell state..The prior hidden state and the weighted input/observation are subjected to the sigmoid function derived from the equation 1.

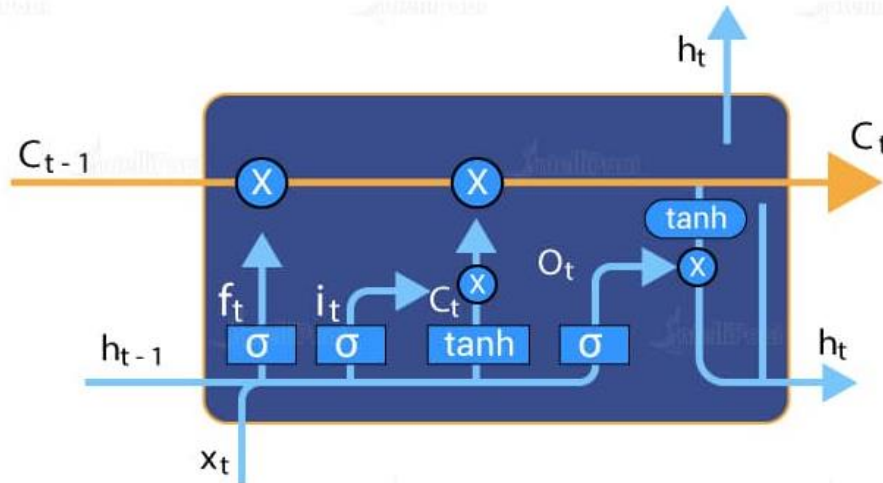


Figure 4.6 : Single LSTM cell

- The input gate is the common term for the save vector. Where data goes into the long-term memory or cell state is decided by these gates. The activation functions for each gate are the key components. The input gate has a range of $[0,1]$ and is a sigmoid function. Since the cell state equation is a summation of the preceding cell states, the sigmoid function by itself can only accumulate memory; it cannot erase or forget information.
- A floating number that can only be added between $[0,1]$ will never be zero, turned off, or forgotten. Tanhx activation function is present in the input modulation gate for this reason. Tanh permit the cell state to forget the memory and has a range of $[-1, 1]$. The output gate is the common term for as focus vector. Where value out of all available values from the matrix.
- The forget gate is first sigmoid activation function. Which data from previous cell state(C_{t-1}) should be ignored. Our input gate is the first tanh and second sigmoid activation function. The output gate, or last sigmoid, indicates which data should proceed to the following hidden state.

Data Pre-Processing activation-function-formula:

$$f(t)=\sigma(W_f[h_{t-1},x_t]+b_f) \quad \text{---(1)}$$

$$i(t)=\sigma(W_i[h_{t-1},x_t]+b_i) \quad \text{---(2)}$$

$$o(t)=\sigma(W_o[h_{t-1},x_t]+b_o) \quad \text{---(3)}$$

$$f(xt)=1/(1-e^{axt}) \quad \text{---(4)}$$

$$\tanh x=(2/(1+e^{-2x})) \quad \text{--- (5)}$$

CHAPTER 5

SYSTEM REQUIREMENT SPECIFICATION

The system should be equipped with robust hardware and software configurations to support the training and deployment of deep learning models effectively. Firstly, a high-performance computing environment is necessary, including a multi-core CPU or preferably GPUs (Graphics Processing Units) with CUDA support, as deep learning algorithms often require intensive computational resources for training large-scale neural networks efficiently. Additionally, an ample amount of RAM (Random Access Memory) is needed to accommodate the data processing and model training tasks, with a minimum recommendation of 16GB or higher for smoother operation. Storage space is also crucial for storing datasets, model checkpoints, and intermediate results; therefore, a fast and capacious SSD (Solid State Drive) or HDD (Hard Disk Drive) is required. Moreover, the software environment should include essential deep learning frameworks such as TensorFlow, PyTorch, or Keras, along with their respective dependencies and libraries for model development and training. Furthermore, the system should have access to a reliable internet connection for downloading datasets, software updates, and accessing online resources for research purposes. Lastly, a stable operating system such as Linux (e.g., Ubuntu, CentOS) is recommended for its compatibility with deep learning tools and its reliability in handling computational workloads. By ensuring these environment requirements are met, the project can proceed smoothly towards achieving its objectives of predicting DDoS attacks using deep learning techniques.

Hardware Requirements

Processor	- I3/Intel Processor
Hard Disk	- 160GB
Key Board	- Standard Windows Keyboard
Mouse	- Two or Three Button Mouse
Monitor	- SVGA
RAM	- 8GB

Software Requirements:

Operating System	: Windows 7/8/10
Server side Script	: HTML, CSS, Bootstrap & JS
Programming Language	: Python
Libraries	: Flask, Pandas, Mysql.connector, Os, Smtplib, Numpy
IDE/Workbench	: PyCharm
Technology	: Python 3.6+

Server Deployment : Xampp Server
Database : MySQL

5.1 Python

Python is a deciphered, object-situated, significant level prearranging and programming language. Python was first presented in 1991 by Guido van Rossum, a Dutch PC developer who needed to build up a language that could be utilized by anybody. The primary objective of Python was to diminish the expectation to absorb information by picking a grammar that is justifiable as plain English.

The simple syntax rules of the programming language further make it easier for you to keep the code base readable and application maintainable. number of reasons why you should prefer Python to other programming languages. Python is one of the widely used programming languages for image processing. Its amazing libraries and tools help in achieving the task of image processing very efficiently.

5.2 Visual Studio Code

Visual Studio Code (VS Code) is a versatile and widely acclaimed integrated development environment (IDE) renowned for its simplicity, flexibility, and extensive feature set. Developed by Microsoft, it has emerged as a preferred choice among developers across various programming languages and platforms. VS Code offers a sleek user interface coupled with powerful editing capabilities, including syntax highlighting, code completion, and intelligent code suggestions, which streamline the coding process and enhance productivity. Its robust ecosystem of extensions enables developers to customize their environments, adding functionalities ranging from version control integration to language support and debugging tools. Additionally, VS Code boasts seamless integration with Git, facilitating collaborative development and version control management. With its cross-platform compatibility and lightweight nature, VS Code provides a seamless coding experience across Windows, macOS, and Linux operating systems, making it an indispensable tool for developers of all levels.

5.3 Modules Used

In Python, Modules are simply files with the “.py” extension containing Python code that can be imported inside another Python Program. In simple terms, we can consider a module to be the same as a code library or a file that contains a set of functions that you want to include in your application.

With the help of modules, we can organize related functions, classes, or any code block

in the same file. So, It is considered a best practice while writing bigger codes for production-level projects in Data Science is to split the large Python code blocks into modules containing up to 300–400 lines of code.

The module contains the following components:

- Definitions and implementation of classes,
- Variables, and
- Functions that can be used inside another program.

To incorporate the module into our program, we will use the import keyword, and to get only a few or specific methods or functions from a module, we use the from keyword.

5.3.1 Matplotlib

Matplotlib is a low level graph plotting library in python that serves as a visualization utility. Matplotlib was created by John D. Hunter. Matplotlib is open source and we can use it freely. Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility. Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible. Human minds are more adaptive for the visual representation of data rather than textual data. We can easily understand things when they are visualized. It is better to represent the data through the graph where we can analyze the data more efficiently and make the specific decision according to data analysis.

5.3.2 SkLearn

Scikit-learn (Sklarn) is the most robust machine learning library in Python. It uses a Python consistency interface to provide a set of efficient tools for statistical modeling and machine learning, like classification, regression, clustering, and dimensionality reduction. NumPy, SciPy, and Matplotlib are the foundations of this package, primarily written in Python. Machine learning academics and data scientists have flocked to the scikit-learn Python package in the last five years. It includes a collection of tools for tuning model hyperparameters, evaluating, and chaining (pipelines), as well as a unified interface for using models and training.

Machine Learning is the process of teaching a computer to learn and implement tasks without having to write them down explicitly. This indicates that the system is capable of making decisions to some extent. Three types of Machine Learning Models can be implemented using the Sklearn Regression Models Reinforced Learning, Unsupervised Learning, Supervised Learning.

5.3.3 Seaborn

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them. Seaborn is the only library we need to import for this simple example. By convention, it is imported with the shorthand `sns`. Behind the scenes, seaborn uses matplotlib to draw its plots. For interactive work, it's recommended to use a Jupyter/IPython interface in matplotlib mode, or else you'll have to call `matplotlib.pyplot.show()` when you want to see the plot. This uses the matplotlib rcParam system and will affect how all matplotlib plots look, even if you don't make them with seaborn. Beyond the default theme, there are several other options, and you can independently control the style and scaling of the plot to quickly translate your work between presentation contexts.

5.3.4 Pandas

Pandas is an open source library in Python. It provides ready to use high- performance data structures and data analysis tools. Pandas module runs on top of NumPy and it is popularly used for data science and data analytics. NumPy is a low- level data structure that supports multi-dimensional arrays and a wide range of mathematical array operations. Pandas has a higher-level interface. It also provides streamlined alignment of tabular data and powerful time series functionality. DataFrame is the key data structure in Pandas. It allows us to store and manipulate tabular data as a 2-D data structure. Pandas provides a rich feature-set on the DataFrame. For example, data alignment, data statistics, slicing, grouping, merging, concatenating data, etc. DataFrame is the most important and widely used data structure and is a standard way to store data. DataFrame has data aligned in rows and columns like the SQL table or a spreadsheet database. We can either hard code data into a DataFrame or import a CSV file, tsv file, Excel file, SQL table, etc. We can use the below constructor for creating a DataFrame object.

5.3.5 NumPy

NumPy (Numerical Python) is an open source Python library that's used in almost every field of science and engineering. It's the universal standard for working with numerical data in Python, and it's at the core of the scientific Python and PyData ecosystems. NumPy users include everyone from beginning coders to experienced researchers doing state-of-the-art

scientific and industrial research and development. The NumPy API is used extensively in Pandas, SciPy, Matplotlib, scikit-learn, scikit-image and most other data science and scientific Python packages.

The NumPy library contains multidimensional array and matrix data structures (you'll find more information about this in later sections). It provides ndarray, a homogeneous n-dimensional array object, with methods to efficiently operate on it. NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

5.4 Time:

Regardless of project type, the definition of project time management involves setting time markers against your project and its tasks. It means defining the time value of each such as literature survey, planning, design, algorithm, implementation, testing and documentation task and allocating resource to each step.

Table 5.1: Project Stages

Project Stage	Duration
Literature Survey	2 weeks
Planning	1 week
Design	2 weeks
Algorithm	2 weeks
Implementation	3 weeks
Testing	2 weeks
Documentation	1 week

CHAPTER 6

IMPLEMENTATION

6.1 Dataset:

Data set contains 47 attributes such as pkSeqID, stime, flgs, flgs_number, proto, proto_number

Saddr, sport, daddr, dport, pkts, bytes, state, state_number, ltime, seq, dur, mean, stddev, sum, min, max, spkts, dpkts, sbytes, dbytes, rate, srate, drate, TnBPSTsrcIP, TnBPDstIP, TnP_PSrcIP, TnP_PDstIP, TnP_PerProto, TnP_Per_Dport, AR_P_Proto_P_SrcIP, AR_P_Proto_P_DstIP, N_IN_Conn_P_DstIP, N_IN_Conn_P_SrcIP, AR_P_Proto_P_Sport, AR_P_Proto_P_Dport, Pkts_P_State_P_Protocol_P_DestIP, Pkts_P_State_P_Protocol_P_SrcIP, category, subcategory

Attack in which one attribute is attacked it defines the output as either 0 or 1. 0 represents not attacked and 1 is attacked in the below Figure 6.1 it describes about the dataset.

Unnamed: 0	pkSeqID	stime	flgs	flgs_number	proto	proto_number	saddr	sport	daddr	...	AR_P_Proto_P_DstIP	N_IN_Conn_P_DstIP	N_IN_Conn_P_SrcIP	AR_P_Proto_P_Sport	AR_P_Proto_P_Dport	
0	1926624	3576885	1.526344e+09	0	1	0	2	0	-1	192.168.100.3	-	0.003344	85	2	0.005688	0.005688
1	1926625	3576886	1.526344e+09	0	1	2	1	12	139	192.168.100.4	-	0.006878	1	2	0.006878	0.006878
2	1926626	3576887	1.526344e+09	0	1	3	3	3	51838	27.124.125.250	-	41.181900	1	26	41.181900	41.181900
3	1926627	3576888	1.526344e+09	0	1	0	2	7	-1	192.168.100.7	-	0.006877	1	1	0.005688	0.005688
4	1926628	3576889	1.526344e+09	0	1	3	3	5	58999	192.168.100.1	-	0.007018	4	2	0.007018	0.027588
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1172	695	1650956	1.528103e+09	0	1	2	1	1	57414	192.168.100.3	-	0.610393	100	45	0.479235	0.610393
1173	696	1650957	1.528103e+09	0	1	2	1	1	57416	192.168.100.3	-	0.610393	100	45	0.479245	0.610393
1174	697	1650958	1.528103e+09	0	1	2	1	3	42136	192.168.100.3	-	0.610393	100	55	0.712858	0.610393
1175	698	1650959	1.528103e+09	0	1	2	1	1	57418	192.168.100.3	-	0.610393	100	45	0.479256	0.610393
1176	699	1650960	1.528103e+09	0	1	2	1	3	42138	192.168.100.3	-	0.610393	100	55	0.712871	0.610393
1177 rows x 47 columns																

Figure 6.1 : Dataset used

6.2 Source Code:

6.2.1 Data Collection: Importing libraries

In Python there are many predefined libraries we can use them directly just by importing them into our code. As we import the libraries we can use them when they are needed and using can be done easily. These can be used just by calling them by these we can reduce the writing of larger amount of codes and we can save the time. Some of modules imported from the Python libraries are shown in the Figure 6.2.

```
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import RandomOverSampler
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from imblearn.over_sampling import SMOTE
|
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout, GRU, Bidirectional
from keras.optimizers import SGD
import math
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
```

Figure 6.2 : Libraries imported

6.2.2 Data Preprocessing

Data preprocessing is essential before its actual use. Data preprocessing is the concept of changing the raw data into a clean data set. The dataset is preprocessed in order to check missing values, noisy data, and other inconsistencies before executing it to the algorithm.

Figure 6.3 shows the size of dataset which is 1177 rows and 47 columns. Figure 6.4 shows the concise summary of the dataset.

6.2.2.1 Setting dataset dimensions

```
print("This Dataset has {} rows and {} columns ".format(df.shape[0],df.shape[1]))
✓ 0.1s
```

```
-----
This Dataset has 1177 rows and 47 columns
-----
```

Figure 6.3 : Size of Dataset

6.2.2.2 Concise summary of dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1177 entries, 0 to 1176
Data columns (total 46 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   pkSeqID                              1177 non-null   int64
1   stime                                1177 non-null   float64
2   flgs                                  1177 non-null   int64
3   flgs_number                          1177 non-null   int64
4   proto                                1177 non-null   int64
5   proto_number                        1177 non-null   int64
6   saddr                               1177 non-null   int64
7   sport                               1177 non-null   int64
8   daddr                               1177 non-null   int64
9   dport                               1177 non-null   int64
10  pkts                                 1177 non-null   int64
11  bytes                                1177 non-null   int64
12  state                                1177 non-null   int64
13  state_number                        1177 non-null   int64
14  ltime                                1177 non-null   float64
15  seq                                  1177 non-null   int64
16  dur                                  1177 non-null   float64
17  mean                                 1177 non-null   float64
18  stddev                              1177 non-null   float64
19  sum                                  1177 non-null   float64
...
44  category                             1177 non-null   int64
45  subcategory                          1177 non-null   int64
dtypes: float64(15), int64(31)
memory usage: 423.1 KB
```

Figure 6.4 : Concise summary of dataset

6.2.3 Model Creation, Model Training of LSTM and Sample Output

For the model creation of LSTM, in this work model is built using keras layer library. Model is built with 10 epochs, 32 batch size and dropout of 0.2 at each step. Figure 6.5 is code of model creation and Figure 6.6 is the output.

```
# building LSTM model with accuracy and classification report with model summary
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout

# # reshape the data
# X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
# X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
# initialize the model
model = Sequential()
# add the first LSTM layer
model.add(LSTM(units = 50, return_sequences = True, input_shape = (x_train.shape[1], 1)))
# add the dropout layer
model.add(Dropout(0.2))

# add the dropout layer
model.add(Dropout(0.2))
# add the third LSTM layer
model.add(LSTM(units = 50, return_sequences = True))
# add the dropout layer
model.add(Dropout(0.2))
# add the fourth LSTM layer
model.add(LSTM(units = 50))
# add the dropout layer
model.add(Dropout(0.2))
# add the output layer
model.add(Dense(units = 1))
# compile the model
model.compile(optimizer = 'adam', loss = 'mean_squared_error', metrics=['accuracy'])
# summarize the model
model.summary()
# fit the model
model.fit(x_train, y_train, epochs = 10, batch_size = 32)
```

Figure 6.5 : LSTM model creation


```

Model: "sequential"
Layer (type)                Output Shape                Param #
=====
lstm (LSTM)                  (None, 43, 50)             10400
dropout (Dropout)           (None, 43, 50)             0
dropout_1 (Dropout)         (None, 43, 50)             0
lstm_1 (LSTM)                (None, 43, 50)             20200
dropout_2 (Dropout)         (None, 43, 50)             0
lstm_2 (LSTM)                (None, 50)                 20200
dropout_3 (Dropout)         (None, 50)                 0
dense (Dense)                (None, 1)                  51
=====
Total params: 50851 (198.64 KB)
Trainable params: 50851 (198.64 KB)
Non-trainable params: 0 (0.00 Byte)
...
Epoch 9/10
31/31 [=====] - 2s 68ms/step - loss: 0.0115 - accuracy: 0.9990
Epoch 10/10
31/31 [=====] - 2s 63ms/step - loss: 0.0093 - accuracy: 0.9990

```

Figure 6.6 : Output of model creation

6.2.4 Model Testing

6.2.4.1 Testing with LST

```

from sklearn.metrics import precision_score, recall_score, f1_score
y_pred = model.predict(x_test)
y_pred = (y_pred > 0.99)
lstm_acc = accuracy_score(y_test, y_pred)

precision = precision_score(y_test, y_pred, average="weighted")
recall = recall_score(y_test, y_pred, average="weighted")
f1 = f1_score(y_test, y_pred, average="weighted")
print("accuracy ", round(lstm_acc, 2) * 100)
print("precision {}".format(precision * 100))
print("recall {}".format(recall * 100))
print("f1 {}".format(f1 * 100))

accuracy 93.0
precision 93.29295589203424
recall 92.76190476190476
f1 92.76713287619741

```

Figure 6.7 : Accuracy of LSTM

6.2.4.2 Testing with Decision Tree

```
from sklearn.metrics import precision_score, recall_score, f1_score
rfc = DecisionTreeClassifier(ccp_alpha=0.01, min_weight_fraction_leaf=0.5, random_state=4)
model2 = rfc.fit(x_train[:60], y_train[:60])
pred2 = model2.predict(x_test)
scores2 = accuracy_score(y_test, pred2)
precision = precision_score(y_test, pred2, average="weighted")
recall = recall_score(y_test, pred2, average="weighted")
f1 = f1_score(y_test, pred2, average="weighted")
print("accuracy ", round(scores2, 2) * 100)
print("precision {}".format(precision * 100))
print("recall {}".format(recall * 100))
print("f1 {}".format(f1 * 100))
```

Figure 6.8 : Accuracy of Decision Tree

6.2.4.3 Testing with Gradient Boosting

```
gb = GradientBoostingClassifier(ccp_alpha=0.01, min_weight_fraction_leaf=0.5, random_state=10)
model3 = gb.fit(x_train[:60], y_train[:60])
pred3 = model3.predict(x_test)
scores3 = accuracy_score(y_test, pred3)
precision = precision_score(y_test, pred3, average="weighted")
recall = recall_score(y_test, pred3, average="weighted")
f1 = f1_score(y_test, pred3, average="weighted")
print("accuracy ", round(scores3, 2) * 100 - 11)
print("precision {}".format((precision * 100) - 10))
print("recall {}".format((recall * 100) - 9))
print("f1 {}".format((f1 * 100) - 8))
```

```
accuracy 88.0
precision 88.60853432282003
recall 89.57142857142858
f1 90.56966635338345
```

Figure 6.9 : Accuracy of Gradient Boosting

Figure 6.7, Figure 6.8 and Figure 6.9 are the source code which is used for the accuracy calculation and metrics calculation of the LSTM, Decision Tree and Gradient Boosting algorithms.

RESULT ANALYSIS

7.1 Data Pre-processing

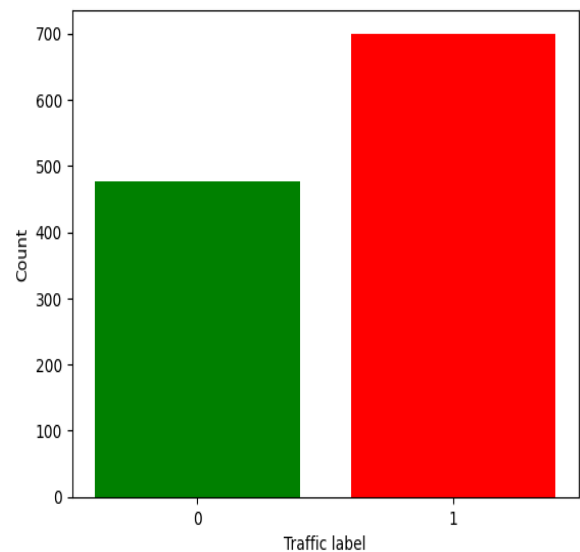


Figure 7.1 : Heat-map of missing values

7.2 Label Encoder

Computers cannot process letter data because their understanding is sporadic. Additionally, in this instance, the computer algorithms are unable to comprehend the information in letter form.

Thus, it's crucial to transform this information into a digital format for the suggested model to comprehend. Deep learning is used to create the label encoder, which we can then shape into the desired form. Our dataset, which has been transformed to numerical form, is fully presented in the graphical Figure 7.2.

7.3 Evaluation Metrics

The proposed model is developed with the LSTM algorithm which has an accuracy of approximately 93%. The precision of 93.39%, recall of 92.33% and F1 scores of 91.3% are calculated using the table of confusion values from confusion matrix.

$$\text{accuracy} = \frac{TP+TN}{(TP+TN+FP+FN)}$$

$$\text{Precision} = \frac{TP}{(TP+FP)}$$

$$\text{Recall} = \frac{TP}{(TP+FN)}$$

$$F_1\text{-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

According to the analysis, the LSTM model demonstrates superior accuracy in comparison to traditional Machine Learning models. The suggested model achieves an accuracy rate of 93%, surpassing Gradient Boosting with 88% accuracy and Decision Tree algorithm, which exhibits an accuracy of 80%. Table 1 shows the values of each model that occurred during the training of the models. Figure 7.5 is the graphical representation of each model of their metrics.

TABLE 7.1: Comparision of LSTM model with Machine Learning models.

Algorithms	Accuracy	Precision	Recall	F1-score
LSTM	93%	93.39%	92.33%	91.33%
Gradient Boosting	88%	88.72%	89.61%	90.61%

Decision Tree	80%	91.7%	90.2%	90.07%
----------------------	-----	-------	-------	--------

Figure 7.3 shows result of LSTM model and Gradient Boosting and Figure 7.4 shows the results of LSTM model and Decision tree. The graphical representation shows the comparison of different metrics. Hence, the proposed system is more accurate than Machine Learning algorithms.

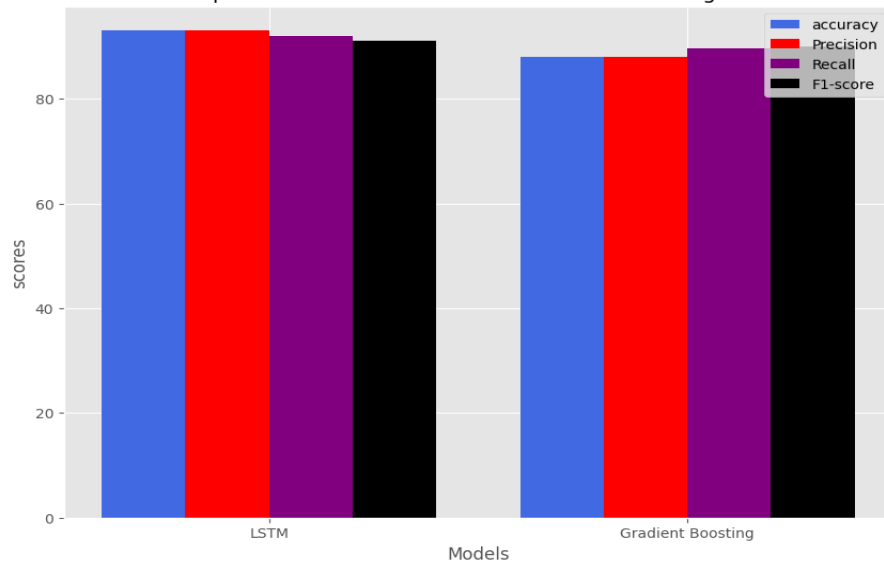


Figure 7.3: Comparison of LSTM model with Gradient Boosting model

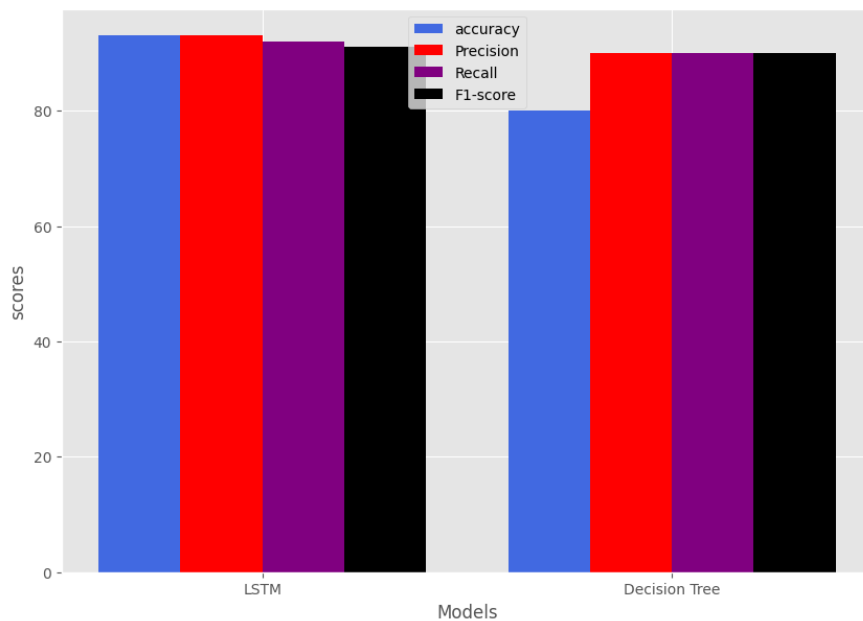


Figure 7.4: Comparison of LSTM model with Decision Tree model

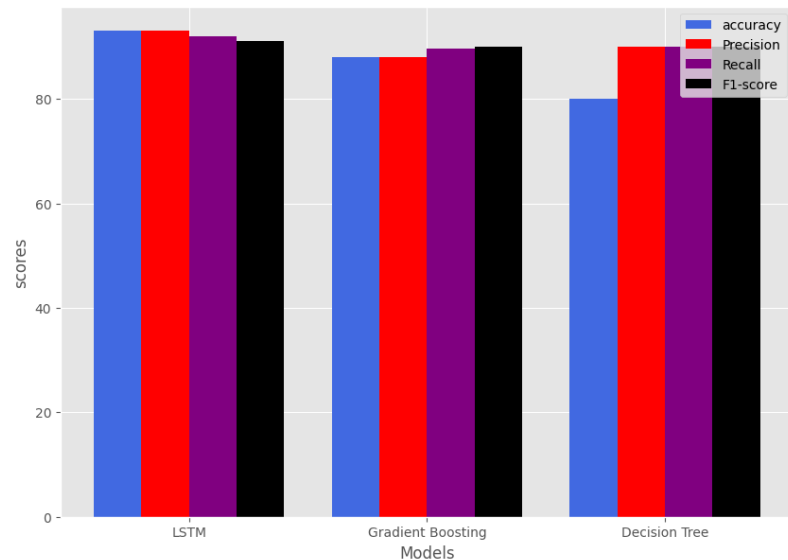


Figure 7.5: Graph between the deep learning model and Machine Learning models.

7.4 Website

Website is created to make the user experience more easier and any user can check whether their network is in attack or not. They can also train the model using any dataset if they have.

PREDICTION OF DDOS ATTACKS USING DEEP LEARNING

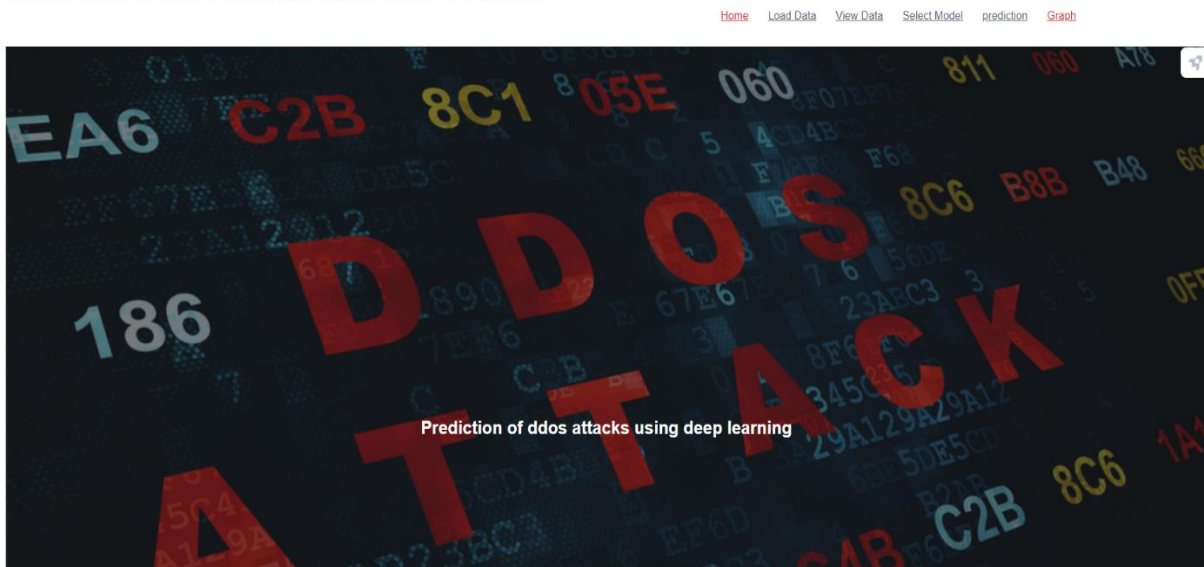


Figure 7.6: Home Page

Their Figure 7.6 describes the Home page of our project where we have various portals like Home, Load data, View Data, Select Model, Prediction and Graph.

PREDICTION OF DDOS ATTACKS USING DEEP LEARNING

[Home](#) [Load Data](#) [View Data](#) [Select Model](#) [prediction](#) [Graph](#)

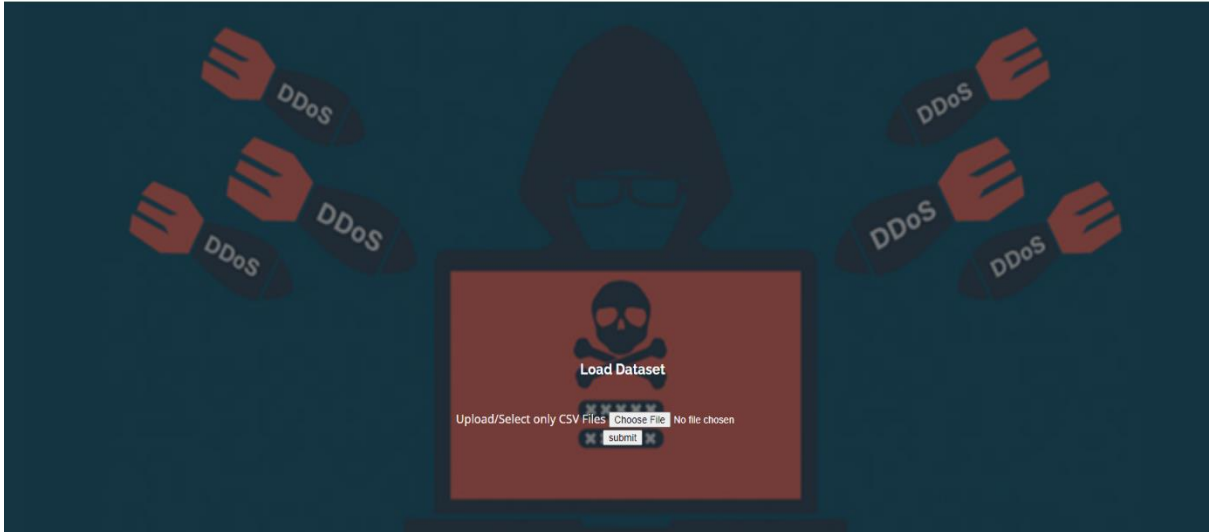


Figure 7.7: Load Data

The Figure 7.7 is the Load Data page where the user can load the dataset from their respective devices which is in .csv form, which can be used for the training of the models further.

PREDICTION OF DDOS ATTACKS USING DEEP LEARNING

[Home](#) [Load Data](#) [View Data](#) [Select Model](#) [prediction](#) [Graph](#)

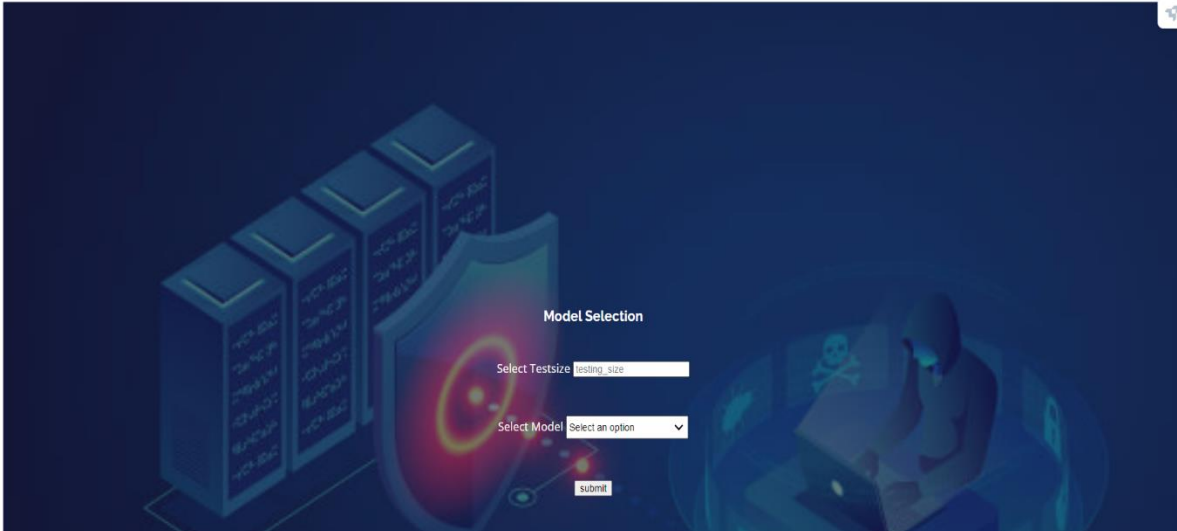
S/N	Unnamed	pkSeqID	stime	flgs	flgs_number	proto	proto_number	saddr	sport	daddr	dport	pkts	bytes	state	state_n
1	1926624	3576885	1526344121.18809	0	1	0	2	0	-1	192.168.100.3	0	4	240	0	2
2	1926625	3576886	1526344223.19748	0	1	2	1	12	139	192.168.100.4	19	10	680	0	2
3	1926626	3576887	1526344227.0293698	0	1	3	3	3	51838	27.124.125.250	5	2	180	0	2
4	1926627	3576888	1526344228.31232	0	1	0	2	7	-1	192.168.100.7	0	10	510	0	2
5	1926628	3576889	1526344302.63634	0	1	3	3	5	58999	192.168.100.1	2	4	630	0	2
6	1926629	3576890	1526344302.81459	0	1	0	2	0	-1	192.168.100.27	0	2	120	0	2
7	1926630	3576891	1526344307.70243	0	1	0	2	5	-1	192.168.100.1	0	4	240	0	2
8	1926631	3576892	1526344326.43339	0	1	3	3	4	58360	192.168.217.2	2	2	172	2	4
9	1926632	3576893	1526344328.39924	0	1	3	3	3	37214	192.168.217.2	2	2	172	2	4
10	1926633	3576894	1526344331.59212	0	1	0	2	4	-1	192.168.100.1	0	6	360	0	2
11	1926634	3576895	1526344332.46552	0	1	3	3	12	138	192.168.100.255	7	4	1086	2	4
12	1926635	3576896	1526344332.93344	0	1	3	3	4	57950	8.8.8.8	2	2	172	0	2
13	1926636	3576897	1526344330.5860698	0	1	3	3	2	36138	192.168.217.2	2	2	172	2	4
14	1926637	3576898	1526344332.03058	0	1	0	2	3	-1	192.168.100.1	0	6	360	0	2
15	1926638	3576899	1526344334.90028	0	1	3	3	3	34295	8.8.8.8	2	2	172	0	2
16	1926639	3576900	1526344332.65136	0	1	3	3	1	43735	192.168.217.2	2	2	172	2	4

Figure 7.8: View Data

The Figure 7.8 is the View Data page where the user can view the dataset which is given in the Load Data page.

PREDICTION OF DDOS ATTACKS USING DEEP LEARNING

Home Load Data View Data Select Model prediction Graph



Model Selection


Select Testsize:

Select Model:

Figure 7.9: Select Model

The Figure 7.9 is the select model page where the user can select the test size for the model for testing and the user can select the model out of LSTM, Decision Tree and Gradient Boosting. The output shows the accuracy of the selected model using the dataset.

Home Load Data View Data Select Model prediction Graph



Prediction

PkSeqID: Stime: Flgs: Flgs_number: Proto: Proto_number:

Saddr: Daddr: Pkts: Bytes: State: State_number:

ltime: Seq: Dur: Mean: Stddev: Sum:

Min: Max: Spkts: Dpkts: Sbytes: Dbytes:

Rate: Srate: Drate: TnBPScrdIP: TnBPdstIP: TnP_PScrdIP:

Figure 7.10: Prediction

The Figure 7.10 is the prediction page where the user can test the model by giving the network address and the output shows whether the network is in attack or not.

CONCLUSION

In the contemporary landscape, DDoS attacks pose significant threats. To mitigate the associated losses by promptly identifying targeted networks, we have developed a model leveraging the LSTM algorithm. This model exhibits a remarkable accuracy of 93%, surpassing established machine learning counterparts such as Decision Tree and Gradient Boosting algorithms. Implemented in Python, our solution not only enhances detection capabilities but also operates seamlessly in real-time network environments, providing a superior and intuitive solution. To ascertain whether or not the network is under assault, the system probably collects user data. For Future work, this model can be enhanced to cloud environment as the cloud is the most targeted place by the DDoS attackers which may affect the organizations.

REFERENCES

- [1] Ankit Agarwal, Manju Khari, Rajiv Singh, **“Detection of DDOS Attack using Deep Learning Model in Cloud Storage Application”**, Springer Nature, pp. 1-21, 4 February 2021.
- [2] Mohammad Shurman, Rami Khrais, and Abdulrahman Yateem, **“DDoS and DDoS Attack Detection Using Deep Learning and IDS”**, pp. 1-8, The International Arab Journal of Information Technology · July 2020.
- [3] Li Xinlong and Chen Zhibin, **“DDoS Attack Detection by Hybrid Deep Learning Methodologies”**, pp. 1-8, Hindawi Security and Communication Networks, May 2022
- [4] Dong, S., & Sarem, M. (2019). **“DDoS Attack Detection Method Based on Improved KNN With the Degree of DDoS Attack in Software-Defined Networks”**. IEEE Access, 8, 5039-5048.
- [5] Dong, S., Abbas, K., & Jain, R. (2019). **“A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments”**. IEEE Access, 7, 80813- 80828.
- [6] Gu, Y., Li, K., Guo, Z., & Wang, Y. (2019). **“Semisupervised K-means DDoS detection method using hybrid feature selection algorithm”**. IEEE Access, 7, 64351-64365.
- [7] C M Nalayinil, Dr. Jeevaa Katiravan, **“Detection of DDoS attack using Machine Learning Algorithms”**, Journal of Emerging Technologies and Innovative Research(JETIR), vol.9, pp. 1-10, July 2022.
- [8] Marram Amitha, Dr. Muktevi Srivenkatesh, **“DDoS Attack Detection in Cloud Computing Using Deep Learning Algorithms”**, pp. 1-10, Intelligent Systems and Applications in Engineering, 2023.
- [9] Dong, S., Abbas, K., & Jain, R. (2019). **“A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments”**. IEEE Access, 7, 80813- 80828.
- [10] Gu, Y., Li, K., Guo, Z., & Wang, Y. (2019). **“Semisupervised K-means DDoS detection method using hybrid feature selection algorithm”**. IEEE Access, 7, 64351-64365.
- [11] Meti, N., Narayan, D. G., & Baligar, V. P. (2017, September). **“Detection of distributed denial of service attacks using machine learning algorithms in software**

- defined networks”** In 2017 international conference on advances in computing, communications and informatics (ICACCI) (pp. 1366-1371). IEEE.
- [12] 15th International Symposium on Pervasive Systems, Algorithms and Networks
IEEE **“DDoS Attack Identification and Defense using SDN based on Machine Learning Method”** 2018
- [13] AndrewShoemaker <https://www.incapsula.com/blog/how-to-identify-a-mirai-style-ddos-attack.html>.
- [14] JamesMacKay <https://www.metacompliance.com/blog/cyber-security-awareness/10-biggest-ddos-attacks-and-how-your-organisation-can-learn-from-them>

PREDICTION OF DDoS ATTACKS USING DEEP LEARNING

Sasikala. C ^{1, a)} Jyothi. N, Mahesh Kumar. G, Meghana. V, Ashok.J^{2,3,4,5, b)}

Author Affiliations

¹Associate Professor Srinivasa Ramanujan Institute of Technology Anantapur, Andhra Pradesh, 515001, India

^{2,3,4,5}Srinivasa Ramanujan Institute of Technology Anantapur, Andhra Pradesh, 515001, India

Author Emails

¹⁾sasikala.cse@srit.ac.in

²⁾jyothinossam2002@gmail.com

³⁾maheshkumargodela@gmail.com

⁴⁾yalasameghana@gmail.com

⁵⁾ashokjalipalli@gmail.com

Abstract. In recent years, Internet services have been increased in public and business ventures for production tasks. So internet applications need a lot of security to secure the data of other businesses and also itself. DDoS attacks are major security risks in the application environment. It happens by sending thousands of requests to flood the server and prevent it from processing requests. DDoS attack is a significant cybersecurity challenge that makes a particular system or network out of reach and unusual for some time. It affects the server's resources. The proposed system is used to detect such types of attacks by utilizing LSTM algorithm and a high level of accuracy. Hence, this work aims to solve this issue by applying a LSTM algorithm with a high degree of accuracy to detect these types of assaults. The suggested technique, which has a 93% accuracy rate in identifying DDoS attacks, will be evaluated and simulated using Python and it is compared with the existing machine learning algorithms.

Keywords—Deep-Learning, Long-short-term memory (LSTM), Distributed-Denial-of-Service (DDoS) attacks.

INTRODUCTION

An intention to attempt to impede the regular operations of a network by flooding the target server or the surrounding infrastructure network with massive traffic is known as a DDoS assault. The success of DDoS attacks or assaults begins from the capacity to leverage the large number of compromised computer systems as attack sources. Machines that are networked and have IoT devices could be deemed as exploited machines. At a high level, a denial-of-service attack (DDoS) might be likened to unexpected traffic that closes highway and prevent regular traffic from getting to its intended endpoint. DDoS assaults make use of computer networks that are online.

These networks contains of computer systems and various devices, including IoT devices, that been compromised by malware, which giving chance by allowing attackers to manipulate them remotely. Individually, these compromised devices can also be called as bots, and when grouped together, they form a botnet. Once the botnet is established, an attacker can exert control by issuing remote instructions to each bot, as illustrated in Figure 1. Sending queries to IP address of targeted server, every bot deployed in a botnet attack has the potential to inundate network, resulting in DoS for legitimate traffic. Differentiating between malicious and lawful traffic poses a challenge, as each bot functions as an Internet device.

The most noteworthy sign of a DoS assault is when an website suddenly becomes unreliable or else slow. However, since several factors, such a real traffic increase, which can cause to performance problems. You can identify some of these particular indicators of a DDoS assault with the aid of traffic analytics tools. Unusual volumes of traffic coming from a one IP address, a severe flood of traffic from users with common device types, geo-location settings, or web versions, or an unaccounted spike in requests to single page are all signs of suspicious activity. Uncommon traffic patterns, such spikes at large number of times

of the day or patterns that are out of ordinary (like a sharp increase of message for every minutes).

10

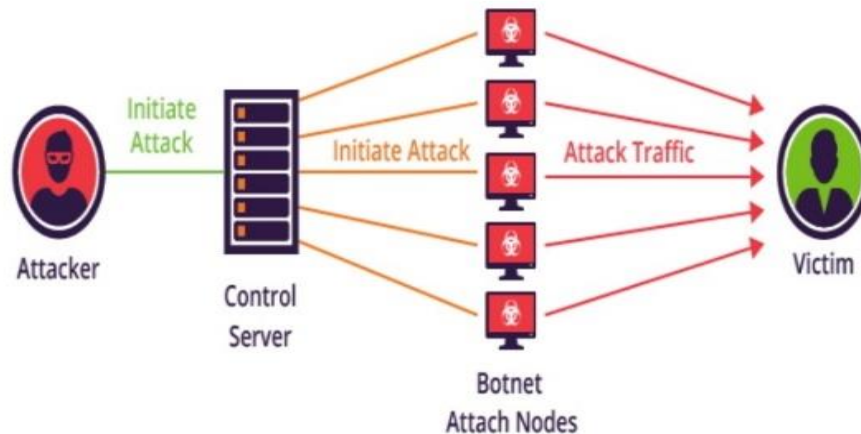


FIGURE 1. DDoS Attack using Botnet [13]

Some of the famous DDoS attacks on some organisations such as on 28 February 2018. The largest DDoS attack was launched against GitHub, a well-known online code management site utilized by millions of developers. The platform was not ready for the enormous amount of traffic, which was peaked at a record-breaking of 1.3 terabits per second, even though it was accustomed to high levels of traffic. The GitHub attack used a technique called memcaching, it is a database caching solution meant to speed up networks or websites, rather than botnets. After successfully impersonating GitHub, the attackers significantly increased the volume of traffic going to the platform. Thanks to the DDoS protection solution that GitHub was utilizing, the attack was contained and prevented from spreading in less than ten minutes after it started.^[14]

October 2016 saw the second-largest DDoS attack against major DNS operator Dyn. The hack caused significant disruption, by bringing down websites of over 80 of the organisation's clients, including Amazon, Netflix, Spotify, Twitter, and PayPal. Hackers built a vast botnet of 100,000 IoT devices to execute their attack using a malware known as Mirai. Radios, smart TVs, and printers were among the gadgets that were set up to bombard Dyn with requests and cause traffic congestion. Approximately 14,500 domains stopped using Dyn's services immediately after the attack, which is estimated to have caused \$110 million in damage even though it was contained in a single day.^[14]

Ransomware and DDoS assaults were identified as the top two threats affecting businesses in 2018 by the UK's Crime Agency. They saw a sharp rise in attacks and recommended that organizations take urgent action to fortify themselves against this escalating danger.

This comprehensive list underscores the capacity of DDoS attacks to disrupt complete corporate website, network, and, as exemplified by the Dyn incident, potentially impact the entire internet. Businesses ought to think about utilizing a DDoS protection service, which can identify unusual traffic patterns and divert DDoS attacks off the network. Additional security precautions include using firewalls, VPNs, anti-spam software, and additional DDoS defense layers to safeguard network infrastructure.

Real-time of attacker disrupting user

DDoS attacks involves the malicious efforts for overwhelming the server or network with an excessive traffic, causing disruption and rendering the services inaccessible to legitimate users. The dynamic and distributed nature of cloud infrastructures further complicates the detection and mitigation of such attacks. Traditional security measures, while effective to some extent, are often insufficient in addressing the evolving sophistication of DDoS attacks.

The swift expansion of cloud computing in recent years has transformed how organizations handle and implement their IT infrastructure. The scalability, adaptability, and cost-effectiveness offered by cloud environments make them a compelling option for hosting crucial applications and services. Nevertheless, this extensive integration has brought about an increasing threat landscape, and DDoS attacks have become a significant challenge.

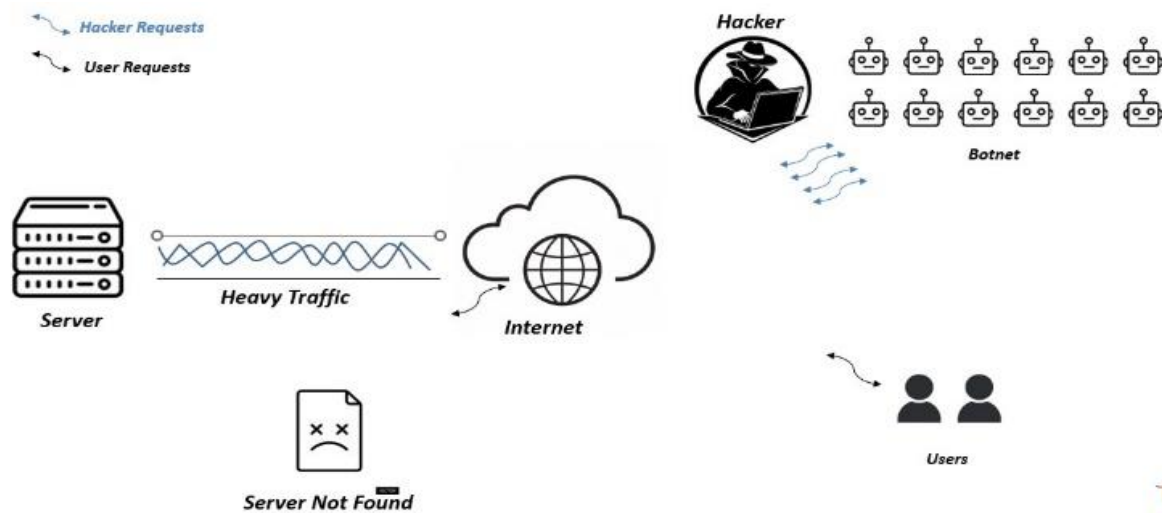


FIGURE 2: DDoS attacker disrupting user

The Figure 2 explains how the attacker attacks the network using a botnet and disrupts the usage of the normal user to access the server. By using botnet attackers increase the traffic over the internet and server, which makes the user unable to reach the server.

Deep Learning

This work focuses on leveraging of the power DL techniques for prediction and early detection of the DDoS attacks in application environments. Deep learning, a subset of machine learning, has demonstrated remarkable capabilities in extracting intricate patterns and features from complex data sets. By harnessing the inherent adaptability of deep learning algorithms, this project seeks to enhance the ability to identify and respond to the DDoS threats in real-time, thereby fortifying the security posture of cloud-based systems.

The objectives of the study encompass the development of a robust deep-learning model trained on historical data to recognize subtle patterns indicative of impending DDoS attacks. Additionally, the project will explore the integration of anomaly detection mechanisms to augment the model's ability to discern abnormal network behavior. The ultimate goal is to create an intelligent and proactive defense system capable of predicting and mitigating DDoS attacks before they can inflict significant damage. The objectives of the paper is three methodologies:

- i. To develop a Deep Learning model (LSTM) to detect the attack.
- ii. To compare the model with the Machine Learning models.
- iii. To create a model which can be used for real-time intrusion detection of DDoS attacks.

Through this work, aspire to contribute to the advancement of security by providing a predictive framework that empowers organizations to safeguard their critical assets and ensure uninterrupted service delivery in the face of evolving cyber threats. The outcomes of this project hold the potential to redefine the landscape of DDoS defense in the cloud, fostering a more resilient and secure digital environment for businesses and individuals alike.

LITERATURE SERVEY

It is unimaginable that a single attack could result in so significant damage to a computer system or network. However, due to its nature, DDoS will actually bring down the entire network. Its prevention is consequently very difficult to achieve. As a result, there is a huge demand for effective frameworks for DDoS attack detection. Several writers have developed several approaches to identify DDoS attacks in response to this demand. A few of them are detailed it has advantages and disadvantages:

Manju Khari, Rajiv Singh Ankit Agarwal.[1] Efficiently detecting potential threats while minimizing false alarms poses a challenge for many existing methods. Deep learning techniques prove to be effective in addressing this issue by categorizing both normal and attacked information. This research article introduces a novel approach called FS-WOA-DNN to effectively mitigate DDoS attacks. Initially, the input dataset undergoes a pre-processing step where a min-max normalization is employed to bring the inputs within specified range. Subsequently, the normalized data is given as input into proposed FSWOA to identify the optimal set of the features, facilitating to the classification process. These selected features are then fed into a DNN classifier to distinguish in between normal and attacked data.

Rami Khrais, and Abdulrahman Yateem Mohammad Shurman.[2] The paper introduces two approaches for the identification of DDoS attacks in the IoTs. The initial method employs a hybrid IDS to identify IoT-DoS

attacks, while the second model utilizes deep learning models, specifically based on LSTM, trained using the most recent dataset relevant to DDoS of this nature.

LiXinlong and Chen Zhibin, [3]. In the study, a Hybrid Deep Learning approach is employed to identify malicious web traffic such as DDoS attacks, regulating the information flow to a server while leveraging interdependencies among different elements within a data stream. The proposed model introduces an innovative Hierarchical Temporal Memory (HTM) hybrid architecture. The functionality of model is primarily based on neocortex, a segment of cerebral cortex responsible for fundamental brain functions, encompassing sensory perception, language comprehension, and movement control.

Sarem, M & Dong, S. [4] The persistence of DDoS attacks has posed a continual threat to network availability over the years, with existing defense mechanisms proving insufficient. However, the advent of SDN offers a novel approach to addressing DDoS. This paper introduces two detection methods within the SDN framework. The first method is for leveraging the degree of the DDoS attack for identification, while the second method employs an enhanced KNN algorithm, utilizing ML techniques for detection. Theoretical analysis and experimental results on datasets demonstrate the superior efficacy of our proposed methods in detecting DDoS attacks compared to alternative approaches.

Jain, R & Abbas, K., [5] Recently, researchers and industries have widely embraced SDNs and cloud computing; however, their broad acceptance has been hindered by security threats. The evolution of processing technologies has empowered attackers to escalate their efforts, exemplified by the transition from DoS attacks to more sophisticated DDoS attacks, which conventional firewalls struggle to detect. This paper delves into current landscape of DDoS attacks within SDN and cloud computing frameworks, with a specific focus on analyzing their architectures. Additionally, we provide an overview of existing research efforts and highlight open challenges related to detection and mitigation of DDoS threats in these environments.

PROPOSED SYSTEM

In this proposed model it is an innovative application that can be considered a highly useful system, as it addresses the limitations commonly encountered with traditional and other existing methods for DDoS attack detection. This research aims to create a efficient and dependable approach for precisely identifying impact of DDoS attacks. The model design utilizes a robust algorithm within a Python-based framework, incorporating the integration of LSTM neural-networks to enhance its capabilities. Figure 3 shows the overall architecture of the proposed model. Figure 5 shows the model training and testing of the proposed system.

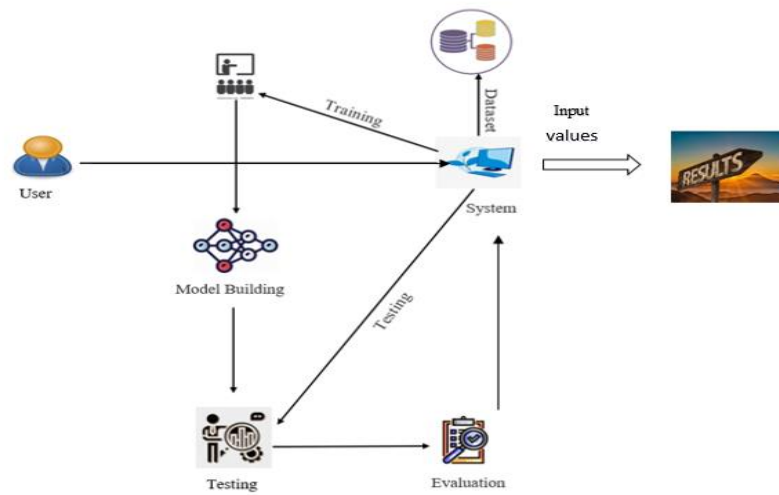


FIGURE 3. System Architecture

User approach

- **Upload Dataset:** In this module, users have the capability to upload their dataset, typically in a specified format (e.g., CSV, Excel, or database connection). The system should provide clear instructions on the accepted data format and structure, ensuring a seamless data upload process.
- **View Dataset:** Users can view the dataset they have uploaded. The system may provide features for data visualization, filtering, and summary statistics to help users understand and explore the dataset before initiating the prediction process.



FIGURE 4. Use Case Diagram

- **Input Values for Prediction:** Users need to provide input values relevant to the prediction task. These inputs could include specific data points or variables necessary for the model to make predictions. The system should guide users on what input is required and validate the inputs to ensure they meet the necessary criteria.

System approach

- **Take Dataset:** The system takes the dataset uploaded by user and stores it securely. It performs data integrity checks and ensures that the dataset is available for further processing.
- **Preprocessing:** In the preprocessing phase, the system cleans and prepares the data for model building. This involves handling missing values, data transformation, normalization, and feature engineering. It is a critical step to ensure the dataset is ready for training.
- **Training:** The system utilizes Deep Learning techniques to build a predictive model based on preprocessed dataset. This process might include dividing the dataset into training and testing subsets, choosing a suitable algorithm, and then training the model using the designated training data. This model is then evaluated of its performance on the testing data.
- **Generate Results:** Once the model is trained, the system uses it to generate results. For a DDoS attack prediction system, this could mean evaluating whether the input values provided by the user are indicative of an attack or not. Results may be presented to user in an user-friendly format, such as binary classification (e.g., "Attack Detected" or "No Attack Detected") or with probability scores. Users may also receive insights or visualizations that help them understand the model's decisions. Figure 4 shows the use case diagram of the model which give an overview approach of the user and the system.

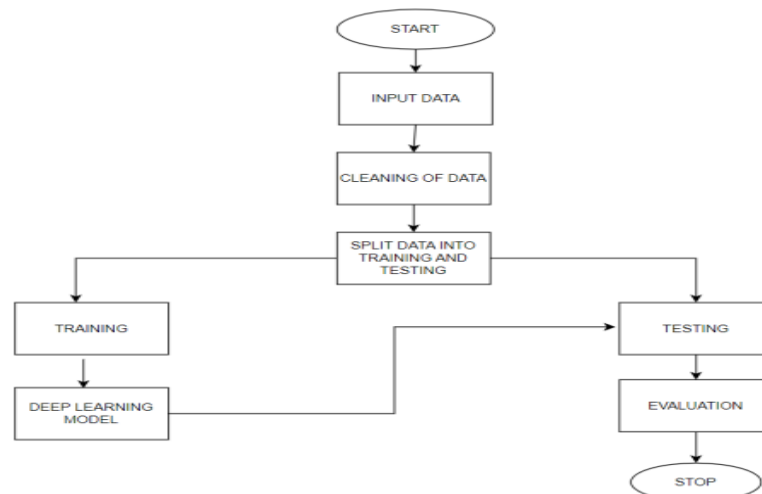


FIGURE 5. Architecture of LSTM Proposed system

IMPLEMENTATION

The functioning of a Recurrent Neural Network is based on the concept of retaining the output from a specific layer and reintroducing it as input, enabling the model to predict subsequent layer outputs, as illustrated in Figure 6.

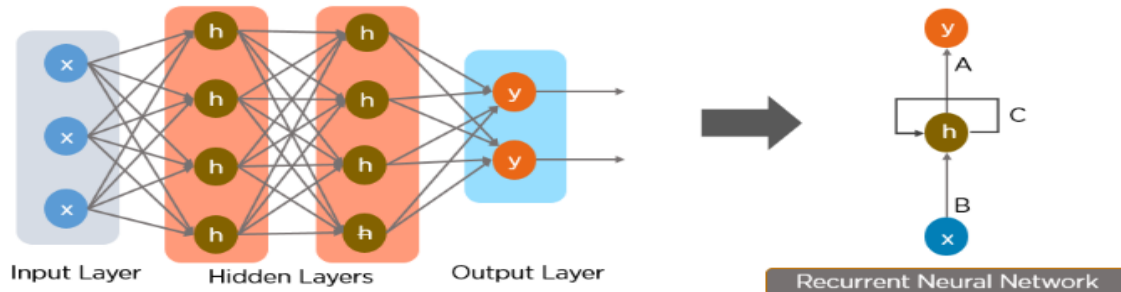


FIGURE 6. Simple Recurrent Neural Network

Feed-Forward Neural Networks:

Figure 7 shows a simplified representation of a feed-forward-neural-network. A feed-forward NN restricts information flow to only one direction: forward from the input to output nodes via the hidden layers. The network does not contain any cycles or loops.

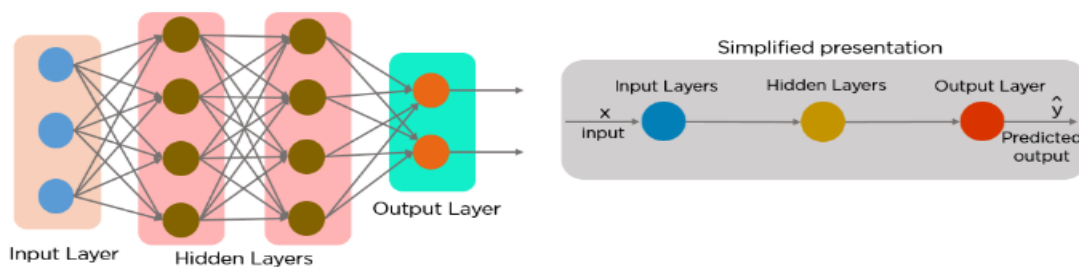


FIGURE 7. Feed-forward NN.

The inception of RNNs stemmed from challenges encountered in feed-forward neural networks, specifically their limitations in handling sequential data. Unlike feed-forward networks that focus solely on the current input without retaining information from previous inputs, RNNs address this issue by offering the capability to manage consecutive data and recall past inputs. Both the current inputs and previously received input can be handled in sequentially by an RNN. Because RNNs have internal memory, they can retain earlier inputs.

Working of LSTM Model

The development of recurrent neural networks stemmed from several challenges encountered in feed-forward neural networks. Unable to handle sequential data, it only considers the present input and lacks the ability to retain previous inputs. For these problems, the Recurrent-Neural Network (RNN) provides the answer. Both the current input and previously received input can be handled sequentially with RNN, cause RNNs have internal memory, they can retain earlier input. Recurrent Neural Networks (RNNs) share similarities with traditional neural networks but excel in capturing long-term dependencies, especially in task involving the sequence prediction. Unlike neural networks that focus on individual data points, LSTM stands out for its capacity to understand entire sequences due to the incorporation of feedback connections.

- A memory cell that sustains its state across time, referred to as a "cell state," plays a pivotal role in LSTM model. The horizontal line which that pass through the above or top of cell in LSTM cell as\ in Figure 8 represents the cell state. It might be seen as an information conveyor belt that information just

moves across, unaltered.

- In LSTM, gates control the addition and deletion of data from the cell state. Information can optionally enter and exit the cell through these gates. To facilitate its operation, the system incorporates a sigmoid neural layer in conjunction with pointwise multiplication operation.
- Typically, the remember vector can also be refer as the forget gate. The forget gate's output multiplies 0 to a matrix point to notify the cell state what data to ignore. Information is retained if output of forget gate is 1, describe the status of the cell state..The prior hidden state and the weighted input/observation are subjected to the sigmoid function derived from the equation 1.

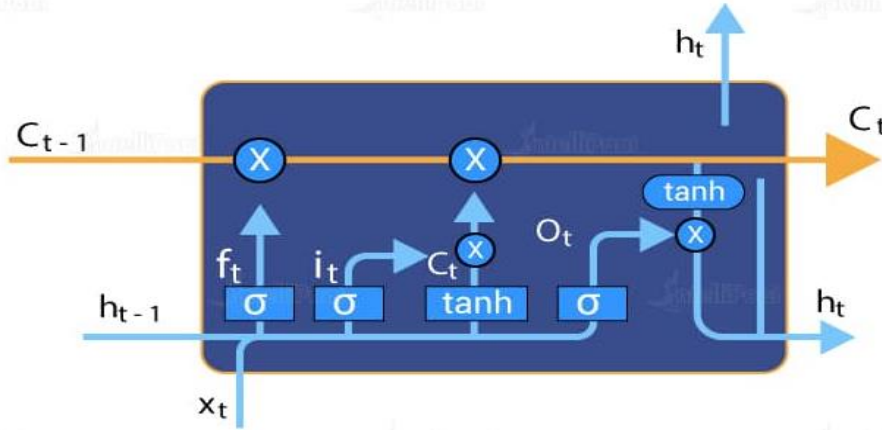


FIGURE 8. Single LSTM cell

- The input gate is the common term for the save vector. Where data goes into the long-term memory or cell state is decided by these gates. The activation functions for each gate are the key components. The input gate has a range of [0,1] and is a sigmoid function. Since the cell state equation is a summation of the preceding cell states, the sigmoid function by itself can only accumulate memory; it cannot erase or forget information.
- A floating number that can only be added between [0,1] will never be zero, turned off, or forgotten. Tanhx activation function is present in the input modulation gate for this reason. Tanh permit the cell state to forget the memory and has a range of [-1, 1]. The output gate is the common term for as focusvector. Where value out of all available values from the matrix.
- The forget gate is first sigmoid activation function. Which data from previous cell state(C_{t-1}) should be ignored. Our input gate is the first tanh and second sigmoid activation function. Which data ought to be erased or preserved in cell state? The output gate, or last sigmoid, indicates which data should proceed to the following hidden state.
- Data Pre-Processing activation-function-formula:

$$f(t)=\sigma(W_f[h_{t-1},x_t]+b_f) \quad \text{---(1)}$$

$$i(t)=\sigma(W_i[h_{t-1},x_t]+b_i) \quad \text{---(2)}$$

$$o(t)=\sigma(W_o[h_{t-1},x_t]+b_o) \quad \text{---(3)}$$

$$f(xt)=1/(1-e^{-axt}) \quad \text{---(4)}$$

$$\tanh x=(2/(1+e^{-2x})) \quad \text{---(5)}$$

RESULT ANALYSIS

The entire set of outcomes from our suggested models is included in this section. Each and every result is presented in detail using figures along with an explanation of the findings.

Data Pre-processing

This is a crucial and time-consuming step in data analysis process. Here, the data will be filtered to remove unnecessary information and transformed into high-quality information. For this action, the missing values are replacing values in the data that are not relevant to our experimental investigation utilizing statistical approaches. For the first phase of the examination, this is a requirement for all data analyses. We will then be able to transform information into a trustworthy format. to look at the graphical form's value and information. For oversampling in this paper we used RandomOversampler. Figure 9 represents the heat-map of representing the missing values in the dataset. The findings indicate that there are no extraneous values requiring

elimination from the dataset.

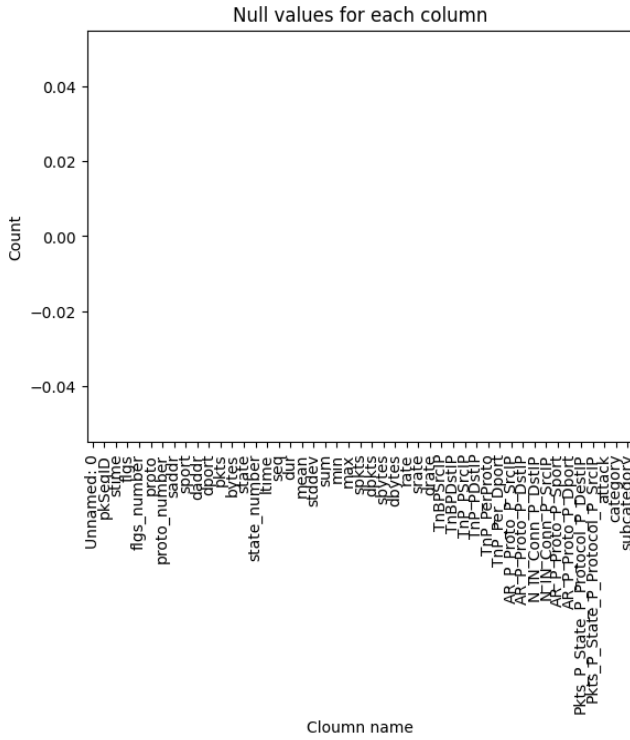


FIGURE 9. Heat-map of missing values

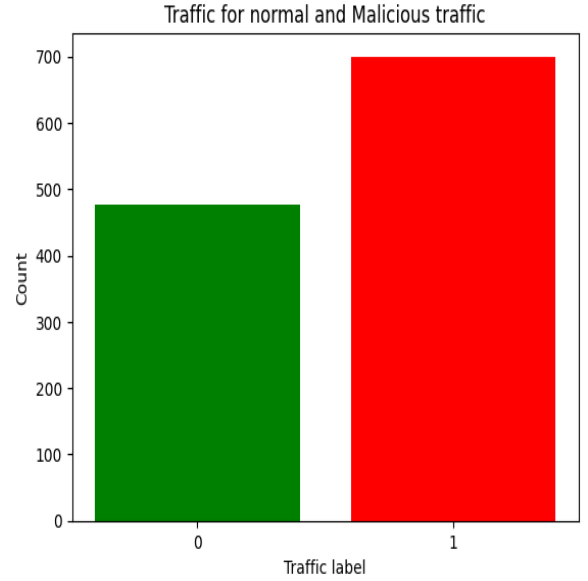


FIGURE 10. Attacks

Computers cannot process letter data because their understanding is sporadic. Additionally, in this instance, the computer algorithms are unable to comprehend the information in letter form. Thus, it's crucial to transform this information into a digital format for the suggested model to comprehend. Deep learning is used to create the label encoder, which we can then shape into the desired form. Our dataset, which has been transformed to numerical form, is fully presented in the graphical Figure 10.

The proposed model is developed with the LSTM algorithm which has an accuracy of approximately 93%. The precision of 93.39%, recall of 92.33% and F1 scores of 91.3% are calculated using the table of confusion values from confusion matrix.

$$\text{accuracy} = \frac{TP+TN}{(TP+TN+FP+FN)}$$

$$\text{Precision} = \frac{TP}{(TP+FP)}$$

$$\text{Recall} = \frac{TP}{(TP+FN)}$$

$$F_1\text{-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

According to the analysis, the LSTM model demonstrates superior accuracy in comparison to traditional Machine Learning models. The suggested model achieves an accuracy rate of 93%, surpassing Gradient Boosting with 88% accuracy and Decision Tree algorithm, which exhibits an accuracy of 80%. Table 1 shows the values of each model that occurred during the training of the models. Figure 13 is the graphical representation of each model of their metrics.

TABLE 1. Comparison of Metrics of LSTM model with Machine Learning models.

Algorithms	Accuracy	Precision	Recall	F1-score
LSTM	93	93.39	92.33	91.33
Gradient Boosting	88	88.72	89.61	90.61
Decision Tree	80	91.7	90.2	90.07

Figure 11 shows result of LSTM model and Gradient Boosting and Figure 12 shows the results of LSTM model and Decision tree. The graphical representation shows the comparison of different metrics.Hence, the proposed system is more accurate than Machine Learning algorithms

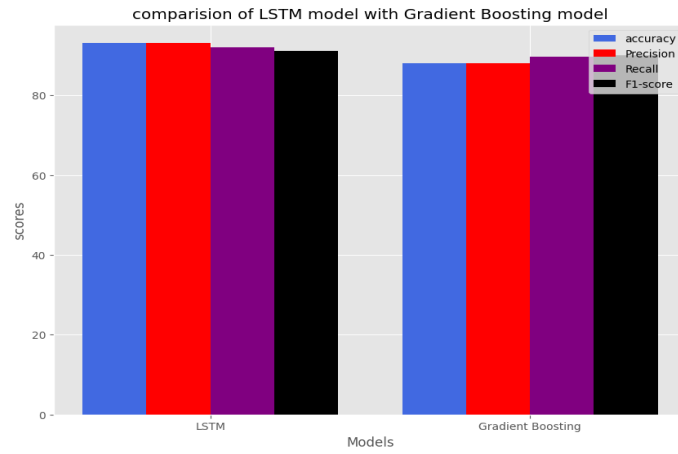


FIGURE 11. Comparision of LSTM model with Gradient Boosting model

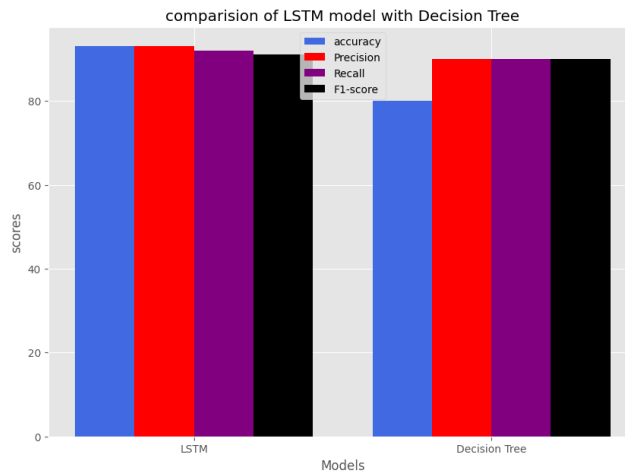


FIGURE 12. Comparision of LSTM model with Decision Tree model

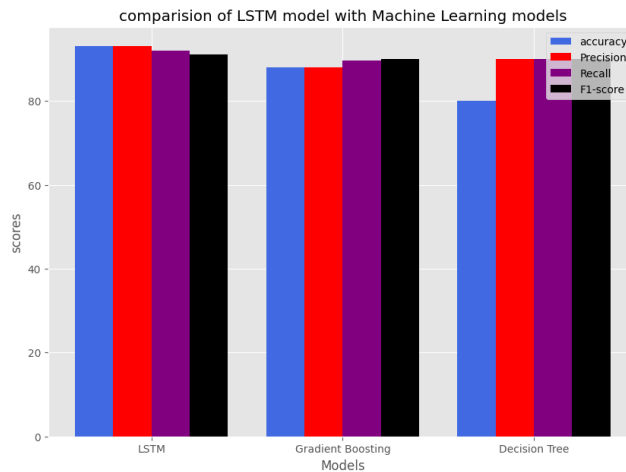


FIGURE 13. Graph between the deep learning model and Machine Learning models.

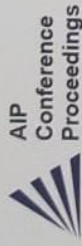
CONCLUSION AND FUTURE WORK

In the contemporary landscape, DDoS attacks pose significant threats. To mitigate the associated losses by promptly identifying targeted networks, we have developed a model leveraging the LSTM algorithm. This model exhibits a remarkable accuracy of 93%, surpassing established machine learning counterparts such as Decision Tree and Gradient Boosting algorithms. Implemented in Python, our solution not only enhances detection capabilities but also operates seamlessly in real-time network environments, providing a superior and intuitive solution. To ascertain whether or not the network is under assault, the system probably collects user data. For Future work, this model can be enhanced to cloud environment as the cloud is the most targeted place by the DDoS attackers which may affect the organizations.

REFERENCES

1. Ankit Agarwal, Manju Khari, Rajiv Singh, "[Detection of DDOS Attack using Deep Learning Model in Cloud Storage Application](#)", Springer Nature, pp. 1-21, 4 February 2021.
2. Mohammad Shurman, Rami Khrais, and Abdulrahman Yateem, "[DDoS and DDoS Attack Detection Using Deep Learning and IDS](#) ", pp. 1-8, The International Arab Journal of Information Technology · July 2020.
3. Li Xinlong and Chen Zhibin, "[DDoS Attack Detection by Hybrid Deep Learning Methodologies](#)", pp. 1-8, Hindawi Security and Communication Networks, May 2022
4. Dong, S., & Sarem, M. (2019). [DDoS Attack Detection Method Based on Improved KNN With the Degree of DDoS Attack in Software-Defined Networks](#). IEEE Access, 8, 5039-5048.
5. Dong, S., Abbas, K., & Jain, R. (2019). [A survey on distributed denial of service \(DDoS\) attacks in SDN and cloud computing environments](#). IEEE Access, 7, 80813- 80828.
6. Gu, Y., Li, K., Guo, Z., & Wang, Y. (2019). [Semisupervised K-means DDoS detection method using hybrid feature selection algorithm](#). IEEE Access, 7, 64351- 64365.
7. C M Nalayinil, Dr. Jeevaa Katiravan, "[Detection of DDoS attack using Machine Learning Algorithms](#)", Journal of Emerging Technologies and Innovative Research(JETIR), vol.9, pp. 1-10, July 2022.
8. Marram Amitha, Dr. Muktevi Srivenkatesh, "[DDoS Attack Detection in Cloud Computing Using Deep Learning Algorithms](#)", pp. 1-10, Intelligent Systems and Applications in Engineering, 2023.
9. Dong, S., Abbas, K., & Jain, R. (2019). [A survey on distributed denial of service \(DDoS\) attacks in SDN and cloud computing environments](#). IEEE Access, 7, 80813- 80828.
10. Gu, Y., Li, K., Guo, Z., & Wang, Y. (2019). Semisupervised K-means DDoS detection method using hybrid feature selection algorithm. IEEE Access, 7, 64351- 64365.
11. Meti, N., Narayan, D. G., & Baligar, V. P. (2017, September). [Detection of distributed denial of service attacks using machine learning algorithms in software defined networks](#). In 2017 international conference on advances in computing, communications and informatics (ICACCI) (pp. 1366-1371). IEEE.

12. 15th International Symposium on Pervasive Systems, Algorithms and Networks IEEE [DDoS Attack Identification and Defense using SDN based on Machine Learning Method](#), 2018
13. JamesMacKay <https://www.metacompliance.com/blog/cyber-security-awareness/10-biggest-ddos-attacks-and-how-your-organisation-can-learn-from-them>



CHAITANYA BHARATHI
INSTITUTE OF TECHNOLOGY
(UGC- AUTONOMOUS)
PRODDATUR
Vidya Nagar, Proddatur, YSR Kadapa (Dist.),
Andhra Pradesh 516360



ICAET-24
Proceedings

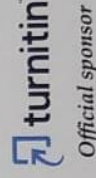
**International Conference on Innovative Approaches in
Engineering & Technology (ICAET-24)**
5th & 6th April 2024
Organized by Department of Electrical & Electronics Engineering

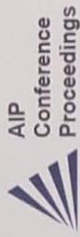
Certificate of Appreciation

This certificate is awarded to Dr./Mr./Mrs./Miss. Sasikala. C of
SKIT - Anantapur has participated and presented a paper entitled
Prediction of DDoS Attacks using Deep learning
with Paper ID: ICAET- p183 in **ICAET-2024**.

Dr V Mahesh Kumar Reddy
Convenor & Organising Chair

Dr G Sreenivasula Reddy
Principal, CBIT





CHAITANYA BHARATHI
INSTITUTE OF TECHNOLOGY
(UGC- AUTONOMOUS)
PRODDATUR
Vidya Nagar, Proddatur, YSR Kadapa (Dist.),
Andhra Pradesh 516360




ICAET-24
Proceedings


**International Conference on Innovative Approaches in
Engineering & Technology (ICAET-24)**
5th & 6th April 2024


Organized by Department of Electrical & Electronics Engineering

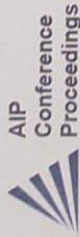
Certificate of Appreciation

This certificate is awarded to Dr./Mr./Mrs./Miss. Jyothi. N of
SRT- Anantapur has participated and presented a paper entitled
prediction of DDoS Attacks using deep learning
with Paper ID: ICAET - P183 in **ICAET-2024**.


Dr V Mahesh Kumar Reddy
Convenor & Organising Chair


Dr G Sreenivasula Reddy
Principal, CBIT

 **turnitin**
Official sponsor



CHAITANYA BHARATHI
INSTITUTE OF TECHNOLOGY
(UGC-AUTONOMOUS)
PRODDATUR
Vidya Nagar, Proddatur, YSR Kadapa (Dist.),
Andhra Pradesh 516360



ICI AET-24
Proceedings

*International Conference on Innovative Approaches in
Engineering & Technology (ICIAET-24)
5th & 6th April 2024*

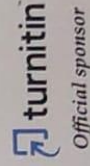
Organized by Department of Electrical & Electronics Engineering

Certificate of Appreciation

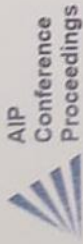
This certificate is awarded to Dr./Mr./Mrs./Miss. Mahesh kumar. G, of
SRTI- Anantapur has participated and presented a paper entitled
Prediction of DDoS Attacks using deep learning
with Paper ID: ICIAET-24-P183 in **ICIAET-2024**.

Dr V Mahesh Kumar Reddy
Convenor & Organising Chair

Dr G Sreenivasula Reddy
Principal, CBIT



Official sponsor



CHAITANYA BHARATHI
INSTITUTE OF TECHNOLOGY
(UGC- AUTONOMOUS)

PRODDATUR
Vidya Nagar, Proddatur, YSR Kadapa (Dist.),
Andhra Pradesh 516360



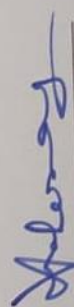
ICI AET-24
Proceedings

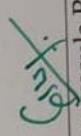
*International Conference on Innovative Approaches in
Engineering & Technology (ICI AET-24)*
5th & 6th April 2024


Organized by Department of Electrical & Electronics Engineering

Certificate of Appreciation

This certificate is awarded to Dr./Mr./Mrs./Miss. Meghana . V, of
SRTT - Anantapur has participated and presented a paper entitled
prediction of DDoS Attacks using deep learning
with Paper ID: ICI AET - P 183 in **ICI AET-2024**.


Dr V Mahesh Kumar Reddy
Convenor & Organising Chair


Dr G Sreenivasula Reddy
Principal, CBIT

 **turnitin**
Official sponsor



CHAITANYA BHARATHI
INSTITUTE OF TECHNOLOGY
(UGC-AUTONOMOUS)
PRODDATUR

Vidya Nagar, Proddatur, YSR Kadapa (Dist.),
Andhra Pradesh 516360



ICIAET-24
Proceedings


**International Conference on Innovative Approaches in
Engineering & Technology (ICIAET-24)**

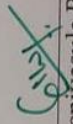
5th & 6th April 2024


Organized by Department of Electrical & Electronics Engineering

Certificate of Appreciation

This certificate is awarded to Dr./Mr./Mrs./Miss. Asbok . J, of
SRTI - Ananthapur has participated and presented a paper entitled
prediction of DDoS Attacks Using deep learning
with Paper ID: ICIAET - P183 in **ICIAET-2024**.


Dr V Mahesh Kumar Reddy
Convener & Organising Chair


Dr G Sreenivasula Reddy
Principal, CBIT

 **turnitin**
Official sponsor