Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
College of Information Technology
Department of Information Networks

Detection and Mitigation of DDoS Attacks in IoT using Entropy and Machine Learning in Fog Computing Environment

A Thesis

Submitted to the Council of the College of Information Technology, University of Babylon in Partial Fulfillment of the Requirements for the Degree of Master in Information Technology / Information Networks

by

Karrar Falih Hassan

Supervised By

Asst. Prof. Dr. Mehdi Ebady Manaa

2022 A.D 1444 A.H

﴿فَتَعَٰلَى ٱللَّهُ ٱلْمَلِكُ ٱلْحَقُّ ۗ وَلَا تَعْجَلْ بِٱلْقُرْءَانِ مِن قَبْلِ أَن يُقْضَىٰٓ إِلَيْكَ وَحْيُهُۥ ۖ وَقُل رَّبِّ زِدْنِى عِلْمًا﴾

صدق الله العلي العظيم
سورة طه اية (١١٤)

Supervisor Certification

I certify that the thesis entitled (**Detection and Mitigation of DDoS Attacks in IoT using Entropy and Machine Learning in Fog Computing Environment**) was prepared under my supervision at the department of Information Networks/ College of Information Technology / University of Babylon as partial fulfillment of the requirements of the degree of Master in Information Technology-Information Networks.

Signature:

Supervisor Name: **Asst. Prof.Dr. Mehdi Ebady Manaa**

Date: / /2021

The Head of the Department Certification

In view of the available recommendations, I forward the thesis entitled "**Detection and Mitigation of DDoS Attacks in IoT using Entropy and Machine Learning in Fog Computing Environment"** for debate by the examination committee.

Signature:

Prof. Dr. **Saad Talib Hasson Aljebori**

Head of Information Networks Department

Date: / /2022

Declaration

I hereby declare that this Dissertation, submitted to the University of Babylon in partial fulfillment of requirement for the degree of Master of Information Technology-Information Networks has not been submitted as an exercise for a similar degree at any other university. I also certify that this work described here is entirely my own except for experts and summaries whose sources are appropriately cited in the references.

Signature:

Name: **Karrar Falih Hassan**

Date: **/ / 2022**

Dedication

I dedicate this thesis

To the soul of my father

To the fountain of patience my mother

To my wife

To my brothers

To my sisters

To my supervisor

To my family

To my friends

Karrar Falih Hassan

List of Publication

No.	Paper Name	Doi
1	Detection and Mitigation of DDoS attacks in Internet of Things using a Fog Computing Hybrid Approach	10.11591/eei.v11i3.3643
2	Detecting Distributed Denial of service Attacks in Internet of Things networks using Machine Learning in Fog Computing	10.1109/IICETA54559.2022.9888699

Acknowledgements

In the name of God, Most Gracious, Most Merciful

At first, greatest praise be to Allah for His assistance in facing the difficulty that I met in my study, and for always helping me to achieve my aims, also for His great graces and boons all the time.

I would like to express my deepest thanks to my supervisor **Asst. Prof.Dr. Mehdi Ebady Manaa**. for his valuable advice, motivation, guidance, and for so many fruitful discussions throughout the preparation of this thesis.

I would like to extend my respect and deepest gratitude to the College of Information Technology and the University of Babylon.

Last but not least, words cannot express my gratitude and indebtedness to my late father, and to my family, especially my sympathetic, compassionate and beloved mother, my dear brothers, my sisters, my faithful wife. Words cannot describe their constant love, care, concern, patience, and direction in every aspect of my life throughout the years of my study. I dedicate the accomplishment of this dissertation to my father's spirit, my affectionate mother, and to the twin of my spirit, my wife.

Karrar Falih Hassan

Abstract

Internet of things (IoT) networks have become more prevalent and widely used, security has become one of the fundamental requirements, and a Distributed Denial of Service (DDoS) attack poses a significant security threat due to the limited resources (CPU, memory, open source, persistent connection) that can be used to either intentionally or unintentionally increase DDOS attacks. A DDoS assault is a congestion-based attack that prevents legitimate users from accessing the victim's resources by flooding them with worthless packets. Detecting a DDoS attack is difficult since there are no standardized guidelines for reliably identifying the legitimate network flow.

In this work, propose an entropy-based framework with machine learning algorithms to optimize the detection result. The proposed system consists of three parts: the first part is to create a sliding time window to calculate the entropy; the second is to use the entropy threshold value to make early detection possible during the DDoS but not after the collapse; The third part is the use of machine learning algorithms (KNN and SVM) to improve the accuracy of the results in early detection.

Several experiments were used to implement the proposed system. The results obtained showed that it is able to detect DDoS attacks with high efficiency. The overall accuracy for CICDDoS2019(Canadian Institute for Cybersecurity Distributed Denial of Service) was 99.99%, the false positive rate was 0, and the log_loss rate was 0.003. While the overall accuracy in the UNSW-Bot-IoT2018 dataset was 100%, the false positive rate was 0 and the log_loss rate was 0. DDoS attack pattern information was extracted from the CICDDoS2019 and UNSW-Bot-IoT2018(University of New South Wales-Bot-Internet of Things) datasets, and the proposed system was trained on them for real-time implementation. The

overall real-time accuracy was 99.88%, the false positive rate was 0.01%, and the log_loss rate was 0.04%.

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

ACKnowledgement	ACK
Application Programming Interface	API
Chain of Custody	CoC
Class of Service	CoS
Central Processing Unit	CPU
Distributed Denial of Service	DDoS
Destination IP	dst_IP
Destination Port	dst_ports
Domain Name System	DNS
Denial of Service	DoS
FINish	FIN
False Negative	FN
False Negative Rate	FNR
False Positive	FP
False Positive Rate	FPR
Graphical User Interface	GUI
Hoeffding trees	HT
Hypertext Transfer Protocol	HTTP
Internet Control Message Protocol	ICMP
Information and Communications Technology	ICT
Intrusion Detection System	IDS
Internet Engineering Task Force	IETF
Internet of Things	IoT
Internet protocol	IP
Internet Research Task Force	IRTF
K-Nearest Neighbors	KNN
Lesser General Public License	LGPL
Linearly Homomorphic Signature	LHS
Mean Absolute Error	MAE
Machine Learning	ML
Network Operating System	NOS
Open Flow	OF
Open Networking Foundation	ONF
Operating System	OS
Random Access Memory	RAM
Support Vector Machine	SVM
Source IP	Src_IP
Source Port	Src_port

Chapter One
Introduction

1.1 Introduction

The introduction of a new technology has aided the exponential growth of IoT, allowing for the connecting of more devices in the IoT network to be made possible by the availability of quicker connections and reduced latency. As IoT networks have become more prevalent and widely, used security has become one of the fundamental requirements.

As technology advances, devices get smaller and less costly, not to mention the widespread acceptance of the always-connected notion in today's networks, the world is becoming more linked. This innovation makes it possible for all devices to connect with one another without difficulty, establishing the framework for the Internet's future [1].

1.1.1 IoT

Improved communication technologies such as 5G, as well as the low cost and high density of sensing devices, have all contributed to the expansion of IoT. IoT applications have the potential to dramatically improve people's lives, including how they live, work, study, and have fun. For example, smart homes, smart health, smart cities, smart agriculture, and other industries may offer occupants a variety of benefits. More items will be linked at a faster rate and with lower latencies as a result of the 5G network, which will improve the efficiency of delay-sensitive Internet of things applications[2]. Figure (1.1) shows the statistics on the growth of the IoT devices up to 2025.

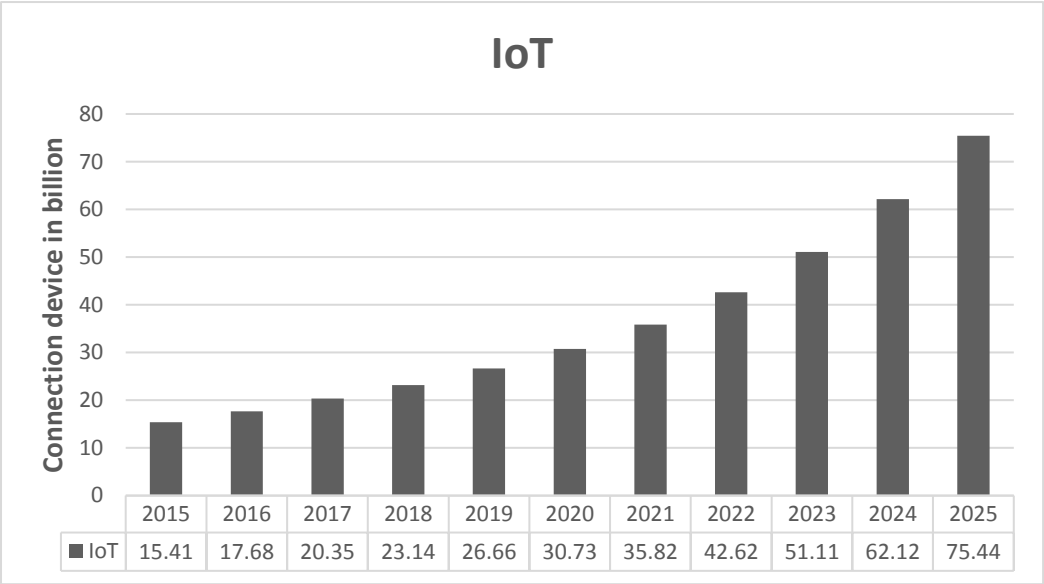

Figure 1.1 IoT connected devices up to 2025[3]

1.1.2 Fog Computing

The proliferation of IoT devices, the amount of data has expanded, resulting in the need for a strong processing engine. In order to get to cloud computing, this vast amount of data must first go via IoT networks, which generates substantial congestion. Eventually, the concept of fog computing arose, which processes data close to the devices and stores it locally in order to reduce overall data flow to the cloud while maintaining high-quality service and giving a speedy, real-time response to applications that demand it[4][5].

When used in conjunction with IoT devices, fog computing can help to alleviate the limitations imposed on the devices by their need for high computing power due to the large amount of data they generate. Data can be sent to the cloud for storage, and the fog layer acts as an intermediary layer between the IoT networks and the cloud layer, allowing for a faster response rate than what was previously available in the cloud.

1.1.3 DDoS attacks

The existence of numerous devices connected to the Internet that launch a DDoS attacks at the same time in order to interrupt the services that the server provides to customers is referred to as DDoS [6]. DDoS attacks are becoming more common. An attacker floods the target server with data packets from network computers to the point that it is unable to receive or reply to data packets, resulting in the system being forced to shut down. However, a significant delay is sufficient to make the system unworkable as a result of the increased reaction time[7].

Because of the fog of computing, it necessitates a continuous Internet connection, making it vulnerable to various types of attacks, such as DDoS, which is one of the most serious types of attacks, in which a large number of requests are made to a server until it stops responding or becomes inaccessible, and which is one of the most serious types of attacks[8].

Because of this growing threat, researcher is coming up with new ways to find and stop attack traffic from botnets on IoT. Machine learning (ML) is a type of artificial intelligence (AI) in which a dataset is used with an algorithm or, in this case, a model to find patterns that can be used to predict what will happen with new data. ML is helpful for this task because it can watch both incoming and outgoing traffic to find traffic that might be suspicious[9].

1.2 Problem Statement

The rise in DDoS attacks, particularly after 2016 and the proliferation of code for the Mirai attack, has made detection and mitigation of DDoS attacks one of the most important issues in recent years:

1- DDOS attacks are most common in IoT devices due to their lack of powerful processing and are unable to ensure security in general [10][11][12].
2- A scarcity of reliable intrusion detection solutions that can provide a secure and acceptable environment for IoT applications [13][14].
3- Most of the solutions offered are trained on legacy datasets that do not include modern DoS and DDoS attacks, so they are not able to accurately detect the attacks[15][16][17].

1.3 Aim and objectives

Aim to robust classifier to detect DDoS in real-time using a proactive and real way. implement detecting mechanisms based on the entropy of network connections and optimize the sliding window time approach for quick entropy computation. Development of an entropy-based detection mechanism using early detection with machine learning algorithms to obtain a classifier capable of detecting attacks with high accuracy.

The following are the primary goals of this study after a review of relevant literature:

1- Securing IoT networks by using machine learning to design a robust system that can stop DDoS attacks in real time.
2- Detecting attacks by collecting traffic using fog computing, processing it using entropy and normalization, and generating a data set to detect the attack early on in its progression.
3- To detect DDoS attacks using a machine learning algorithm with minimal computer resources (CPU and RAM usage).
4- Evaluating the method by applying a performance criterion that takes into account accuracy, precision, Brier score loss, recall, f1-score, mean absolute error, Geometric Mean, Cohen's Kappa,

Matthews correlation coefficient throughput and root mean square error.

5- Contrast the results with previous work.

1.4 Related works

This section is meant to give light to various studies that are related to the one that is being discussed here. A significant number of researchers make use of symmetric algorithms in order to detect DDoS assaults using an algorithm that combines data mining and machine learning.

Neural networks and KNN may be used to identify DDoS in IoT traffic with high accuracy, based on IoT-specific network patterns, such as limited endpoints or packet-to-packet jitter. Doshi et al.,2018, KNN algorithm achieved 99.9% for (TCP, UDP and HTTP)DDoS attacks[18].

With the aid of the Support Vector Machine (SVM) algorithm, Gurulakshmi and Nesarani, [19],2018, suggested the analysis of IoT Bots against DDoS attacks in order to categorize the traffic against regular and abnormal flow, as well as to forecast early aberrant activity. The KNN algorithm achieved 97% and SVM 98% low accuracy rate as compared with others.

According to Adeilson et al., [20],2018, Complex Event Processing (CEP) technology, which permits real-time analysis and processing of data to identify DDOS assaults in edge computing to achieve quick reaction time, has been suggested. The suggested system Achieve 93.10 for syn flood and 99.24 for UDP flood attack, but the fraction of missing data exceeds 8%, which is irrational given the high accuracy acquired.

Rajesh Kumar et al., [21],2018, developed an SVM model to gather assaults on Internet of Things devices while employing the Cowrie honeypot as a collection tool. They classified the attacks into different types using machine learning algorithms (namely, Naive Bayes, JRandom Forest, J48 decision tree, and SVM) and evaluated its performance using machine learning algorithms (namely, Random Forest, Naive Bayes, J48 decision tree, and SVM) with accuracy ranging from 67.7% to 97.39%.

A methodology for detecting DDoS assaults in fog computing is proposed by Joao Vitor Cardoso et al.,2019, utilizing a Raspberry Pi 3B as a fog server. DoS and DDoS assaults are generated with the help of the HPING3 application (SYN Flood, Ping Flood, and UDP Flood). They conducted a simulation to determine the resources required to prevent DoS and DDoS assaults, and the findings revealed that the capacity to identify and block the addresses of attackers took less than 20% of the CPU and 1% of the RAM, respectively[22].

A fog computing approach was used by Luying Zhou et al., [14],2019, to mitigate DDOS attacks in three stages: the first stage involved a firewall capable of filtering botnet attacks in real time, the second stage involved using network functions to analyze traffic, and the third stage involved central coordination of connecting local servers to cloud services. The results showed that the detection rate of TCP protocol type attacks is 99.56% and Modbus is 70.35% in the Fog level alone, while the detection rate of TCP protocol type attacks is 99.84% and Modbus is 88.02% in the Fog computing approach. The results showed that the detection rate of TCP protocol type attacks is 99.84% and Modbus is 88.02% in the Fog computing approach.

Using fog computing for quick and precise detection, Muhammad et al.,2020, offer a framework for mitigating DDOS assaults on IoT. A database that maintains signatures of previously identified assaults CICDoS 2019 was utilized in conjunction with skew-based mitigation, which use the k-NN classification method to identify DDOS attacks. They said that the model would be able to identify DDOS assaults with a high degree of accuracy 99.99%[23].

DDoS attacks were detected in two parts by Shubham Bishnoi et al., [24],2021, using deep learning in a fog environment. The first part used LSTM (long short-term memory) to classify the attacks into two categories, benign and harmful; the second part used CNN (convolutional neural networks) to classify the data, with accuracy reaching 98% in both parts.

Vimal Gaur and Rajneesh Kumar,[25],2021, used a hybrid methodology to identify features. The researchers reduced the number of features to 15 and obtained a high accuracy of 98.34% when using the CICDDoS2019 dataset.

Churcher et al., [26],2021, proposed multi ML methods, including NB, RF, SVM, DT, ANN, KNN, and LR, to highlight their applications and uses in IDS. They draw a comparison between machine learning techniques for binary and multi-class classification on the Bot-IoT dataset, not implementation on real time.

Tong Liu et al.,[27], 2021, Deep learning technology may reliably identify distributed denial of service attacks when combined with blockchain technology. This is accomplished by incorporating an AE and MLP into the deep learning system. The researchers tested their method on three different kinds of datasets and found that it had an accuracy of F1-

score of up to 95% for all three types (BoT-IoT, CIC- IDS2017, and CICDDoS2019).

Rami and Ahmed ,[28], 2021, employed a number of different machine learning algorithms and compared them to one another in order to decide which sort of algorithm was the most effective while they were evaluating this algorithm utilizing CICDDoS2019 datasets. Their method, which was based on several classifications, was able to attain varying degrees of accuracy depending on the kind of algorithm it used.

Table 1.1 A Recapitulation of the Related Works

Author	years	Method	Advantage	Disadvantage	ACC.
[18]	2018	KNN, SVM, DT, RF and ANN	High accuracy rate	The accuracy of the data should be calculated when it is both normal and abnormal	KNN=99.9%
[19]	2018	KNN and SVM	Implementation in real time	Low accuracy rate	KNN=97% SVM=98%
[20]	2018	CEP	Good accuracy rate	percentage of missing data reaches 8%	93.10% for syn and 99.24 for UDP attacks
[21]	2018	NB, J48, DT, RF and SVM	Multiclass detection	Low accuracy rate and high time	SVM=97.39%
[22]	2019	-	Reduce resource use	Not mentioning the methods used for detection	-
[14]	2019	NFV	High accuracy rate	Low accuracy rate Modbus protocol	99.84 for TCP attack

[23]	2020	k-NN	High accuracy rate	Only detection method	KNN = 99.99%
[24]	2021	LSTM and CNN	Using two forms in real time	Low accuracy rate	98%
[25]	2021	KNN and XGBoost	Extract the best feature	Low accuracy rate	-
[26]	2021	KNN,SVM,DT,NB,RF and ANN	High accuracy rate	The accuracy of the data should be calculated when it is both normal and abnormal	KNN=99%
[27]	2021	AE and MLP	Combined Deep learning with blockchain	Not implement in real time	F1=95%
[28]	2021	KNN,SVM,NP,DT, and RF	Multi classification	Low accuracy rate and not implement in real time	-

In light of the aforementioned studies, entropy appears to be effective in the DDoS detection situation, but striking a balance between efficiency and accuracy remains a difficult task. On the basis of this, the study presented in this paper makes an additional progress.

1.5 Thesis Outline

In addition to the first chapter, the suggested system is separated into four more chapters.

Chapter two: This chapter discussed the concepts of IoT , DDoS attack, IoT architecture, Datasets, Machine learning and evaluation.

Chapter three: It was discussed in this chapter how to implement the suggested system, how to demonstrate the system's practical

phases, and how to explain the proposed algorithmic system inside IoT in DDoS detection.

Chapter four: This chapter describes the findings and provides an evaluation of the system technique that was used.

Chapter five: The conclusion of the findings was reported in this chapter. It also included a description of future works.

Chapter Two
Theoretical Background

2.1 Introduction

IoT is a network of connected devices, including smart appliances and sensors, that are able to exchange information with one another over the internet. IoT has applications in a variety of fields, including smart cities, smart homes, and so on[29].

The concept of IoT was initially put out by Kevin Ashton in 1999, and it has since grown in popularity. Aiming to connect everything, everywhere, at any time, IoT is predicted to connect billions of devices in the near future[30].

Because of the rapid development of the IoT, cloud computing, edge computing, and other sophisticated technologies in recent years, an increasing number of resource-constrained intelligent Internet of Things devices have begun to appear in people's daily activities as a result of the rapid development of these technologies. The fact that these Internet of Things devices have limited resource capabilities does not detract from their ability to perform a broad variety of activities via the interchange of data, collaboration, communication, and collaboration with edge and distant servers[31].

It is possible that the volume of data generated by millions of networked devices may be overwhelming[32]. For real-time applications, fog computing has the potential to outperform traditional hardware configurations and software tools since they are incapable of processing and analyzing massive volumes of data in a timely way[33]. A consequence of this is that fog computing acts as a connection between the cloud and Internet of Things devices, enabling real-time analytics of data that cannot be handled on the end device while also allowing data from

several devices to be connected together and evaluated as a single entity. In spite of the fact that the computing and storage capabilities of edge devices are often limited, they are capable of processing data as soon as it is recognized and making speedy decisions independent of data from other devices[34].

2.2 The architecture of IoT

There are many architectures of IoT devices, each of which will be explained separately:

2.2.1 Human brain Internet of Things architecture

There is no general consensus on the design of IoT, and academics have suggested numerous architectures. [35] suggested an architecture for IoT inspired by the human brain for its capacity to think and make choices fast. The suggested architecture consists of three parts as shown in Figure (2.1): -

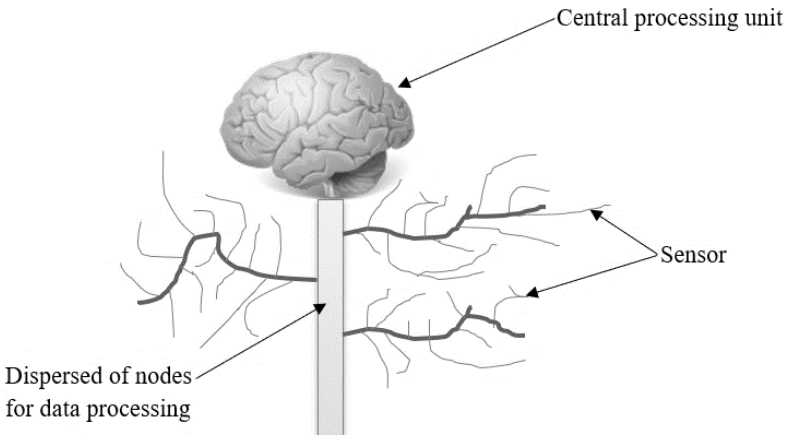

Figure 2.1 human brain Internet of Things architecture[35]

i. **The human brain**: It is akin to the central processing unit for data.
ii. **Spinal cord**: a set of nodes dispersed for data processing.
iii. **Nerve network**: to interact and be compatible with the gadgets of IoT and the network.

When it comes to IoT, experts have separated the levels of the network into three and five layers[11-12],

2.2.2 Three-layer architecture

with the three-layer design being classed as starting from the simplest structures as illustrated in Figure (2.2). The following structure encapsulates the basic concept of IoT architecture:

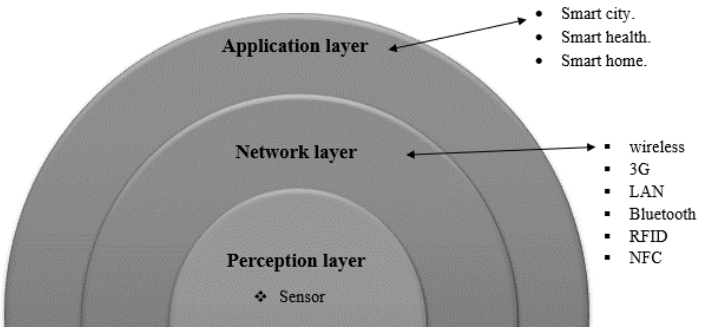

Figure 2.2 three-layer Internet of Things architecture[11-12]

i. **Perception layer**: This layer includes the sensors, which are believed to be the physical layer through which the environment may be perceived or even the devices in close proximity to it can be detected or recognized.

ii. **Network Layer**: This layer manages the connection between Internet of Things devices, servers, and network devices. It allows data to be transported for processing and storage between devices and between servers and network devices.

iii. **The application layer**: is near to the user since it offers him with a variety of application services such as health care, smart cities, and other similar services.

2.2.3 Five-layer architecture

as depicted in Figure (2.3), provides subtle aspects of searching in IoT because, in addition to the perception layer and application layer, it provides us with three additional layers (transport, processing, and business layers): -

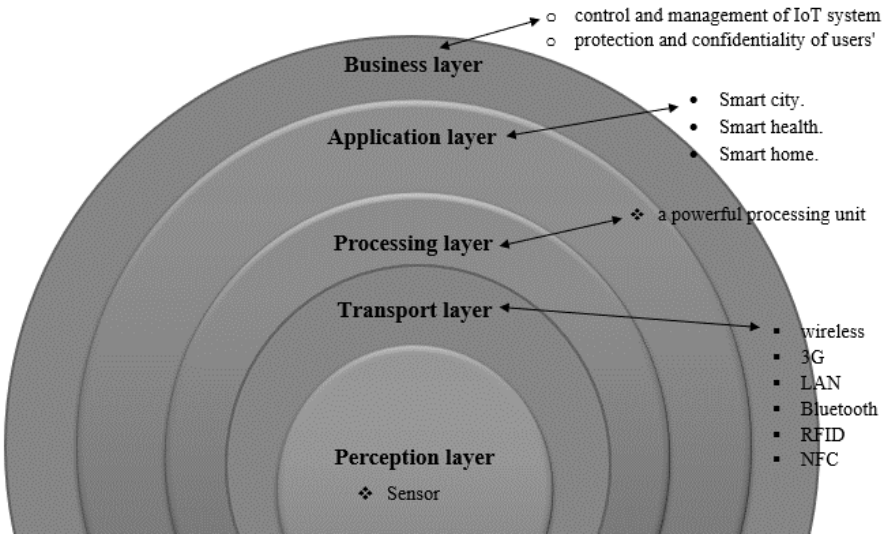

Figure 2.3 five-layer Internet of Things architecture [11-12]

i. **Perception layer**.

ii. **Transport layer**: The process of transporting sensor data from the perception layer to the processing layer or vice versa over a network is referred to as the "transport layer," and the phrase "transport layer" is used interchangeably.

iii. **Processing layer (middleware layer)**: This layer makes use of a sophisticated processing unit in order to handle and store the vast amounts of data that it gets from the transport layer. The transport layer sends the data to this layer so that it may be processed.

iv. **Application layer**.

v. **Business layer**: via it, can control all aspects of IoT system, including the protection and confidentiality of users' data and information.

2.3 DDoS attack

The following is an explanation of what is meant by the phrase "explicit attempt to prohibit legitimate users from accessing the service in question" when applied to the context of a DoS attack: This objective may be accomplished with the use of a DDoS attack by simultaneously launching a large number of entities designed to disrupt service on a computer network[38]. In its simplest form, it is malicious conduct intended to obstruct the flow of workflow traffic via a server, network, or other piece of hardware. The main goal is to swiftly shut down the system's infrastructure and prevent a flood of data from entering it. This form of attack has the potential to be effective when the devices and systems that it targets have been hacked.

This breach, which occurs in a dispersed environment, where data transmission occurs among diverse network devices that are scattered throughout the network without the ability to exercise centralized control, is often referred to as a DDoS assault [39]. In order to carry out a DDoS attack, the attacker must obtain control of the network and the devices that help in its execution. Hackers can take over the machine with the use of malicious software. Each bot is remotely controlled by hackers, who then guide it to the IP address of the desired source.

Hackers transmit hundreds of orders to the robots that are outfitted with the malware, causing a port or server to become overloaded

as shown in Figure (2.4). The DDoS assault aims to make the service unavailable for regular traffic, which is the goal of the attack [40].

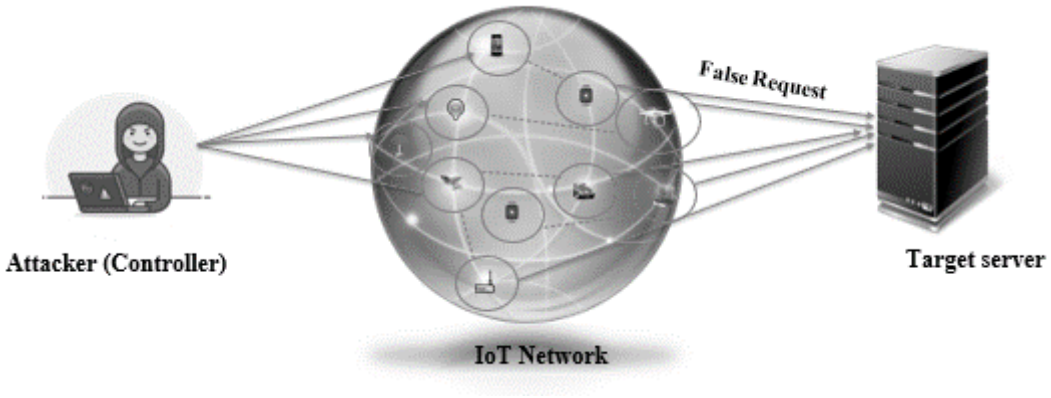

Figure 2.4 DDoS attacks in IoT network

IoT which was once considered a blessing to technology, is now on its way to becoming a curse by contributing to massive DDoS attacks. As more and more things are connected to the internet, there is a greater possibility that they will be targeted in a DDoS attack. These assaults are carried out on the basis of malware infections, which are dispersed across the network as a consequence of numerous IoT devices being breached and taken over by the attackers.

These breaches are the result of numerous IoT devices being taken over by the attackers. The strength of such an assault is directly proportional to the number of compromised Internet of Things devices that the attacker uses to initiate the attack. Bots are the term used in the hacking community to refer to these exploited IoT devices[41]. The following are examples of botnets that have recently been discovered:

A large number of interconnected IoT devices provide a favorable environment for DDoS assaults; as a result, malware implementation bots may be deployed on IoT quite simply:[42]

2.3.1 Mirai

Mirai is one of the malwares that is gaining notoriety these days as a consequence of its indisputable influence on the 2016 DDoS assault. It is meant to infect Linux PCs and IoT devices, and it is one of the malwares that was responsible for the attack[43]. Malware written for Linux has been responsible for the transformation into bots of millions of computers running Linux and other Internet of Thing's devices.

2.3.2 WireX Botnet-2017

The command-and-control protocol used to infect Android devices includes its name as one of the delimiter strings. It had featured thousands of Android cellphones that were running apps that had the impression of being real but were, in fact, malicious malware. These programs provided the appearance of being authentic but were designed to steal information.

2.3.3 Reaper Botnet:

This bot has the capability to search the devices that are part of IoT for flaws and weaknesses that might allow an attacker to take control of such devices.

2.3.4 3ve-2018

WhiteOps, the FBI, Google, and other ad fraud-fighting businesses recently took down 3ve, the most sophisticated digital ad fraud scam, in the autumn of this year[44].

2.4 DDoS Attack Classification at the IoT Layer

The Observation Layer, Network Layer, and the Application Layer are the three fundamental tiers that make up IoT [45]. DDoS assaults vary depending on which layer is targeted, as shown in Figure (2.5):

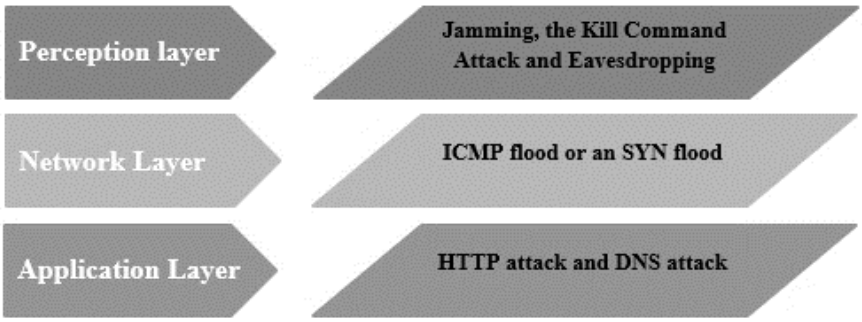

Figure 2.5 DDoS Attack Classification at the IoT Layer

2.4.1 Network Layer

Similarly, to TCP/network IP's layer, this layer has the same security issues that influence confidentiality, availability, and integrity of data as the network layer[46]. The network layer, which includes both wired and wireless networks, is the most vulnerable to assaults, as large amounts of data are used to carry out the attack. The system that gathers the data tries to delay responding to requests and allocating the necessary resources if there are no more direct connections left, which eventually makes it impossible to deliver the service:

An ICMP flood or an SYN flood assault are two examples of network layer attack[47].

2.4.2 Application Layer

The most basic user interface, which may be found in mobile applications, the web, and other places, is contained in the application

layer. The application layer is also the layer through which applications are utilized. A reprogramming attack and a path-based DoS can occur in this tier[48].

The following are some well-known DDoS assaults that have grown increasingly popular in recent years:

i. Flooding of ACK and SYN messages

In order to carry out an ACK flood attack, attackers take advantage of TCP when the three-way handshake process is in progress[49]. The sending of a SYN packet is the first step in the three-way handshake protocol that will take place. The device to which the connection needs to be made then sends a (SYN + ACK) packet in response. In turn, it is responded to by the ACK packet, which marks an end to the connection setup phase. The device then responds by sending (the SYN + ACK) packet [50] as shown in Figure (2.6). When an attacker has malicious intent, he sends an ACK-enabled packet with a spoofed source address, which is often ignored by the target device since it does not have an established connection to the host with the spoofed IP address. This necessitates the processing of each incoming packet, depleting available resources in the process.

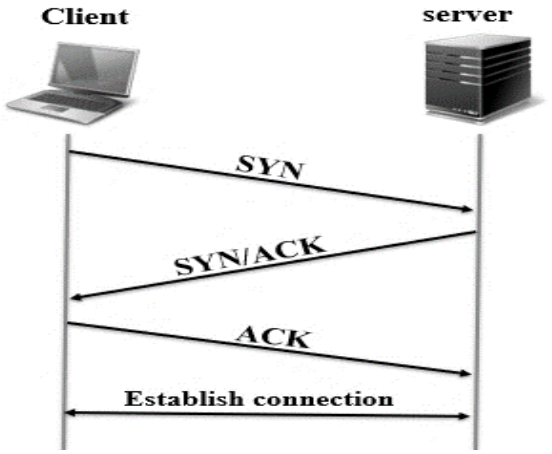

Figure 2.6 TCP three-way handshake [49]

The SYN flood attack is a form of assault in which the attacker sends a high number of SYN packets to the device that is the target of the attack in order to begin the process of the three-way handshake[51]. The targeted device sends back a response consisting of a SYN + ACK packet, which he reads as a confirmation of his request. The whole connection is only partially opened since the attacker purposefully forgets to deliver the necessary ACK packet. So, the targeted server has to wait for an ACK packet from the right source address before it can finish setting up the connection.

ii. DNS (Domain name server) amplification attack

The DNS amplification vulnerability is used in this attack by the UDP amplification vulnerability. In this attack, the UDP application responds with data that is multiplied in contrast to the initial request that prompted the responses. To be more specific, it takes use of a mechanism known as reflection, in which the attacker poses as an administrator of the target system while sending a DNS query to a recursive DNS server., as seen in Figure (2.7) [52].

Figure 2.7 DNS amplification attack

The query will receive a response from the recursive DNS server, which will include the answer to the question. An attacker will carry out an amplification attack by sending a large number of DNS queries to the recursive DNS server. Each of these queries will ask for all of the records associated with a particular domain, but they will do so from a variety of spoofed IP addresses. The victim's network becomes overloaded as a result of the additional traffic, which also renders the DNS infrastructure unavailable. The DNS query packets are designed to create a response packet that is much larger than the query packet as a result, which results in the packets being amplified in this specific way. The main goal of this attack is to send several large response packets to the victim's network, which will eventually cause the victim's network to become overloaded[53].

iii. UDP (Uniform Datagram Protocol) flood attack

By taking this strategy, the intruder will endeavor to maintain their anonymity while simultaneously attempting to continuously broadcast UDP packets to random ports on the target device. If such an application does not already exist on the target device, then the device will be forced to investigate each port and send an ICMP destination unreachable message if it discovers a problem with any of them. In the event that the target device does not include such an application, it will respond with an unavailable packet.

As a consequence of this entire process, the gadget in question will not be used since there will be an excessive amount of waiting[54].

iv. HTTP flood attack

A distributed denial-of-service attack is carried out in this attack by the cybercriminal by taking advantage of valid HTTP GET or POST requests in order to initiate the assault[55]. These attacks need less bandwidth to be shot into the targeted server than other attacks do due to the fact that they do not involve spoofing or reflection techniques.

v. ICMP (Internet Control Message Protocol) flood attack

It frequently generates error messages to alert the source of any network or destination issues, such as when a data packet cannot reach its destination because the gateway is down or when a packet cannot be accessed by the destination. These failures can occur for a number of reasons, including: when a data packet is unable to reach its destination due to a failure of the gateway; when a data packet is not accessible by its Ping is a function that may be carried out in ICMP, and when it is, it causes an echo request to be broadcast and then receives an echo reply in response.

If you do not receive this answer, it is an indication that the other server is not operational or that the other host does not have the capability to send a ping request[56]. In this circumstance, echo requests continue to be made without waiting for an echo response, causing the network to become clogged with traffic and waste unnecessary bandwidth. A smurf assault[57] is another term for this type of attack.

2.5 Machine learning

Humans have used a wide variety of tools to do a wide variety of jobs in a more efficient manner. The inventiveness of the human brain resulted in the development of a variety of devices[58]. These technologies make human existence easier by enabling individuals to satisfy a variety of

living demands, such as transport, industry, and computing, all at the same time. And machine learning is one of those technologies[59].

Several areas, including social networking services, data mining, traffic forecasts, and medical diagnostics, have profited from machine learning. There are three types of challenges that Machine Learning may address: classification, regression, and clustering concerns. Depending on the types and categories of readily accessible training data, it may be necessary to select the most appropriate machine learning algorithm from among the available techniques, such as supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning [60].

From the foregoing, it was discovered that distributed denial-of-service assaults adhere to a pattern while they are active. In this context, the significance of machine learning algorithms in effectively dealing with patterns is highlighted. Machine learning's main goal is to find patterns in data so that decisions can be made based on this data[61].

2.6 Machine learning algorithm

The kind of algorithm employed depends on the kind of problem you wish to solve, the number of variables, the kind of model that would suit it best and so on. Here's a quick look at some of the commonly used algorithms in machine learning (ML)

2.6.1 KNN

Classification strategies are used in the process of doing research to develop a model or function that explains and characterizes the ideas or data classes for a certain set of requirements. Many approaches or procedures are used in classification, and one of them is KNN method, which is used for categorizing objects based on learning data, with the

closest distance to the object being used as the classification criterion. Actually, classification refers to the attempt to forecast which cases will fit into a given group or class, as opposed to regression, which is concerned with predicting the number of values a variable will have[62].

The KNN classification approach is one of the most extensively used classification techniques. It is often utilized because of its simplicity of understanding and short calculating time. The selection of the parameter k is quite important in this procedure. In order to access the training error rate and validation error rate on distinct k values, it is necessary to access both parameters at the same time.

The KNN technique may be used to classify text across a wide range of topics and domains. Machine learning methods that are either predictive or descriptive have been successfully applied to the resolution of a wide variety of problems. These methods include data mining, natural language processing, picture identification, and expert systems, amongst others[63].

Nonparametric, instance-based, or lazily applied: The KNN algorithm is one of the most widely used methods in data mining and machine learning, as well as one of the simplest[64].

The KNN approach is predicated on the concept that the most similar samples that belong to the same class have a high probability of being picked. This is the fundamental premise of the KNN method. The query is predicted based on which of the k nearest neighbors have the major class in the training dataset, as is common for KNN methods. This is done after the technique identifies which of the k nearest neighbors have the query's main class in the training dataset. As a direct consequence of this,

it was only just recognized as one of the best 10 data mining algorithms[65].

The KNN approach is usually sensitive to the choice of the k value, in addition to being widely acknowledged for its effectiveness. In spite of the fact that studies on this topic have been carried out for a significant amount of time, calculating the K value for the KNN algorithm continues to be a challenging endeavor[66].

[67] a said that while selecting an acceptable value for k, it should be able to meet the equation k = $\sqrt{n}$ for training datasets that have a sample size of more than 100 individuals. [68] researched a Bayesian method to assist us in making the right decision in selecting k as the major variable.[69] Despite the fact that there are no theories to ensure that k = $\sqrt{n}$ is appropriate for each test sample,[70] As previously stated, it has been demonstrated that a fixed k value is incompatible with a large number of test samples in a given training dataset.

In Figure (2.8), it can be seen that the border becomes smoother as the value of k is increased. The most difficult aspect of the KNN algorithm is determining the appropriate value of k. In the case when k is equal to 1, the procedure is referred to as the closest neighbor algorithm, for short. The KNN method is incredibly simple to construct and requires just two parameters to function properly.

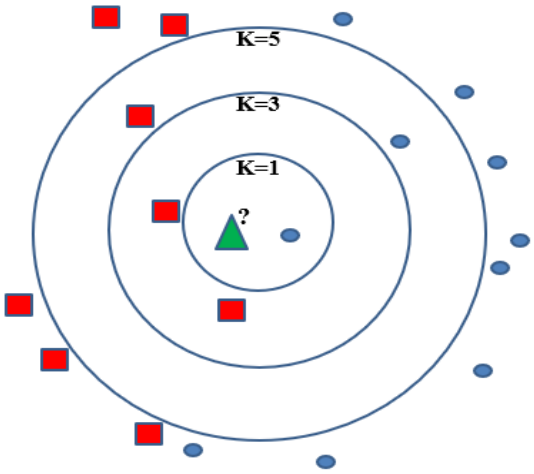

Figure 2.8 KNN classification is shown in this simple example. Test sample green triangle is put in the blue circle group if k = 1, if k = 3 then the red quadrilateral group, and if k = 5 the red quadrilateral group.

The description of the KNN algorithm, as well as the well-studied and sluggish learning approach[63], were both investigated. The performance of the classifier is only dependent on two parameters: k and the degree of similarity or dissimilarity between two classes. In addition, the KNN classification is employed for the test set derived from the training set. For each row of the test set, the k training set vectors that are closest to it using the Euclidean metric are computed. This is done for each column of the test set. It is categorized based on the votes cast by a majority of the judges, with any ties being resolved at their discretion. If there are any ties for the k-the closest vector, all of the candidates are included in the voting process for that candidate.

The various distance metrics are discussed in further detail in the following paragraphs[71]:

i. **Euclidean distance**

It is the most often used and default distance in most situations.is given by[72] and be computed using Equation(2.1)the distance between two subjects on a level with a variety of options a(x1, x2) and b(y1, y2)

$$Dist = (a, b) = \sqrt{(x1 - y1)^2 + (x2 - y2)^2} \quad (2.1)$$

X1 and x2 is point that want to calculated the distance with y1 and y2)

ii. Manhattan Distance

It computes the differences in the coordinates of a pair of data points by comparing them.

$$Dist = (a, b) = |x1 - y1| + |x2 - y2| \quad (2.2)$$

X1 and x2 is point that want to calculated the distance with y1 and y2)

iii. Chebyshev distance

Specifically, the Chebyshev distance is a vector space metric defined as follows: The distance between any two vectors is equal to the sum of their differences along any coordinate dimension. It was given this name in honour of Pafnuty Chebyshev.

It is sometimes referred to as the Tchebychev distance or the maximal metric.

$$Dist = (a, b) = MAX(|x1 - y1|, |x2 - y2|) \quad (2.3)$$

Algorithm 2.1 Pseudo code of a KNN algorithm in general [73]
1- start
2- Load the data.
3- Set the value of K to its initial value.
4- Taking into account each point in the testing set consists of:

Calculate the distance between the testing data and all rows of the training set using the (Euclidean distance, Chebyshev distance, and Manhattan Distance) formula.

4.1. Sort the distances that have been calculated in ascending order based on the distance value.

4.2. Choose the first K rows from the sorted array.

4.3. Find the class that appears the most often in these rows.

4.4. Return the projected class label.

5- End.

2.6.1.1 KNN algorithm has a number of Pros and Cons[74]**:**

Pros:

i. Simplicity in terms of comprehension and application

ii. Training that is quick.

iii. It has a high sensitivity to noisy training data.

iv. It is beneficial when a sample of several different class labels is used.

Cons:

i. It is characterized by laziness in learning.

ii. It is sensitive to the structure of the data on the local level.

iii. The cost of memory.

iv. It operates at a sluggish pace since it is a supervised, slow learner

2.6.2 SVM

In the late 1970s, Vapnik and his team created the support vector machines methodology. Since then, a variety of machine learning applications have employed it as one of the most well-liked kernel-based learning methods[74].

The input data is converted into a high-dimensional space using kernel techniques (functions), which are then trained and returned to the two-dimensional space, which is excellent for two-class scenarios in which both positive and negative samples are present. The goal of SVM analysis is to find a separation plan, also known as a hyperplane in the feature space, that optimally separates the two sample classes (Figure 2.9)[75].

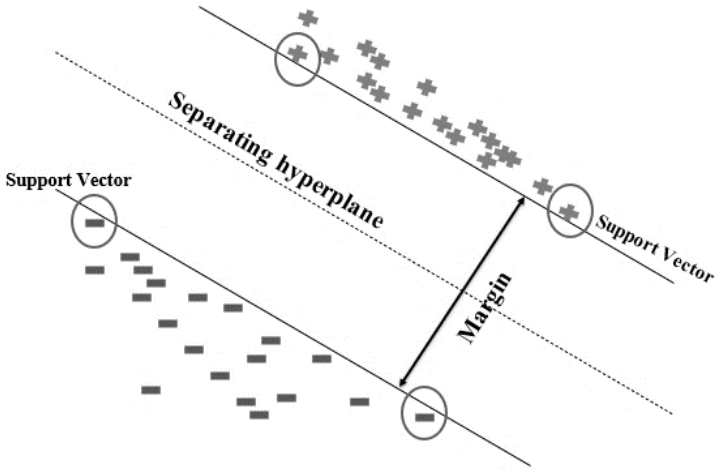

Figure 2.9 SVM classification sample two classes

There is a lot of overlap between the data samples of different classes in practice, in this case, linear SVM cannot guarantee a high level of classification accuracy and requires changes. Soft margin and kernel trick approaches were proposed by Cortes and Vapnik[76] to alleviate the limitations of linear SVM. In order to handle data that is not linearly separable, it may be necessary to include additional components (also known as slack variables) while doing SVM optimization using the soft margin approach. Because of this kernel approach (in which the feature space is transferred into a higher dimension, such as Euclidean or Hilbert space), it is now much simpler to differentiate between the various categories[77].

To optimize SVM performance, choose a kernel function that produces dot products in the higher-dimensional feature space[78]. This space, which may have an unlimited dimension and in which linear discrimination is possible, may have an infinite dimension in theory. Several kernel models, including the Sigmoid, Radial basis function(rbf), Polynomial(poly), and Linear, can be utilized to create SVMs that satisfy Mercer's requirement. Models such as the linear, rbf, and poly are included in this category.

In the field of remotely sensed image processing, the polynomial kernel and rbf kernel are two of the most often employed kernels[79]. The vast majority of the time, kernels are selected by first defining the model of the kernel (whether it be Gaussian, polynomial, or anything else), and then adjusting the kernel parameters through the use of tuning procedures, which may be fairly computationally intensive. The most important factor to consider when selecting a kernel function for classification is the performance of the classifier on a subset of the training sample or the validation set.

The performance of an SVM technique is significantly impacted by the kernel function as well as other hyper-parameters, and the performance of the whole model may be improved by fine-tuning these parameters. These parameters are as follows[80][75]:

i. **Kernel functions**:

The functions that are used by the kernel are referred to as "kernel functions." In the field of machine learning, the term "kernel" refers to a linear classifier that is utilized in order to address a problem that is not linear. It makes use of a space that was originally low-dimensional but has

now been converted into a space that is high-dimensional. A kernel function is represented mathematically by the equation Equation(2.4),

$$K(x, y) = \{f(x), f(y)\} \quad (2.4)$$

x and y are considered to be inputs with an n-dimensional dimension, K is the kernel function, f is a map from an n-dimensional space to an m-dimensional space, and (x, y) represents the dot product of (x) and (y). (x and y) are considered to be n-dimensional inputs. SVM algorithms frequently make use of kernel functions such as the 'linear', 'polynomial', 'radial basis function (RBF)', 'Gaussian', and 'sigmoid' functions. Other kernel functions include the 'radial basis function (RBF)'. RBF and polynomial kernel functions are typically advised responses in large-scale modelling (LSM) inquiries. In contrast, linear kernel functions are better suited for basic data.

ii. **penalty function or regularization parameter (C-value)**:

controls the cost-benefit ratio between it and a less capable hyperplane. It does this by lowering the total amount of mistakes, which in turn enhances the misclassification in each training sample. If the C-value is too high, it may cause the data to be over-fit, despite the fact that hyper-plane separation has the potential to be useful in some circumstances. On the other side, in the event that the C-value is too low, there is a possibility that the data will exhibit misclassifications or under-fitting.

iii. **Gamma**:

This value is known as the kernel's coefficient. When the value of gamma is high, the fit to the training data is closer to perfect, and the opposite is true when the value is low.

iv. **epsilon insensitive loss function (The ε-value):**

The function's ε-value influences its accuracy. A small value increases the number of support vectors, whereas 0 causes overfitting and a big value decrease them.

v. **Degree**:

is used to referring to the order of a polynomial in the polynomial kernel function, which is used to locate the hyperplane that will split the data.

Algorithm 2.2: Pseudo Code for SVM algorithm in general [81]

Start

Require: A and B are loaded with training data that has been labelled,

α <= 0 or α <= partially trained SVM

1. C <= some value (30 for example)
2. **Repeat**
   2.1.for all {ai, bi}, {aj, bj} do
   2.2.Optimize the values αi and αj
   2.3.end for
3. **Until** no change in α

Ensure αi > 0

4-Labal data

End

2.7 Dataset

utilize the two largest and most up-to-date DDoS datasets publicly available online. believe that a comprehensive range of DDoS attack classes, both from reflection-based and exploitation-based DDoS attacks, contained in these datasets are relevant to the Smart Transport Systems context and aligns well with the types of DDoS attack [82]. Use

CICDDoS2019 [83] that contains the largest number of DDoS attack classes that are up to date. BoT-IoT dataset contains three different types of attacks but again only extracted DoS/DDoS attack-related data samples. These datasets have two different types of sub-datasets, one set is used in the training phase while this other set is used in the testing phase of a machine learning model.

2.7.1 CICDDoS2019 Dataset

For this investigation, the CICDDoS2019 dataset that was compiled by the Canadian Institute for Cybersecurity at the University of New Brunswick was utilized. This comprehensive dataset includes 50 million occurrences, 80 characteristics, and 11 different class labels for the purpose of DDoS attack forecasting.

2.7.2 IoT-Botnet UNSW-2018

In the Cyber Range Lab in the core of UNSW Canberra Cyber, where the BoT-IoT dataset was developed after being collected, a realistic network environment was constructed. The environment contains both conventional Internet traffic and botnet traffic. The dataset's source files are provided in many formats, including the csv files and etc. It contains three types of distributed denial of service attacks TCP, HTTP, and UDP.

extract Five points from this database (Source IP/port, Destination IP/port and Protocol) and then transform them using entropy and make them into four points and store them in a CSV file. So, train machine learning algorithms to predict future attacks.

2.7.3 Feature selection

The selection of features is one of the primary ideas in machine learning, and it has a substantial impact on the overall performance of models. A model's performance for the identification of DDoS attacks might be significantly impacted by characteristics that are inappropriate or

only partly connected. In the process of model creation, the first and most significant phase should be the cleaning and selection of the data features. The process of deciding which features will be included in a product, either manually or automatically, is known as feature selection. The existence of unsuitable features in the data might cause a reduction in the quality of the models and cause them to train based on unimportant characteristics.

The two feature selection methods are filter and wrapper models. The filter technique chooses characteristics based on a threshold and user requirements. Feature importance can be assessed individually and by analyzing the whole feature space. The filter technique accepts continuous and categorical information and generates categorical and continuous replies. Filter techniques have a significant association between independent variables. This variable independence must be considered when training a model. High-accuracy wrapper approaches exist for dealing with these difficulties. Searching for a subset of features, creating a machine learning model, and assessing model performance is part of the wrapper technique. This technique is repeated until the goal is reached. This repetition makes wrapper techniques costly. An embedded technique is provided to overcome the drawbacks of filter and wrapper methods. Various filter and wrapper approaches increase accuracy to do this.

2.7.4 Calculating entropy to identify DDoS attacks

To detect DDOS assaults, it is necessary to measure the randomness of the network using the entropy. The notion of entropy was introduced for the first time by the scientist Claude Shannon in his article "A Mathematical Theory of Communication" published in the year 1948 [84]. This is known as the entropy principle. It is possible that the entropy is large, which suggests that the random distribution is high, but it is also possible that the entropy is zero, which implies that all values are the same.

This idea may be used to identify DDoS assaults based on the parameters of the packet (sourceIP,destinationIP,sourcePort,destinationPort) and the calculation of randomness. A collection of packets' entropy is calculated, with the number 1 indicating that the randomness is high and 0 indicating that the packets are all identical to one another. The general formula of Entropy:

$$H(X) = -\sum_{i=1}^{n} p(x_i)\log_2\left(p(x_i)\right) \quad (2.5)$$

Where: H(X)= entropy of X, X= discrete random variable, $P(x_i)$= possible outcomes (x_1 to x_n), which occur with probability $P(x_1)$ to $P(x_n)$. Algorithms that use machine learning look for patterns in data by comparing and contrasting the similarities and differences among different data points. There is an issue, however, when the traits are measured on such vastly different scales.

2.8 Evaluation Metrics

A confusion matrix is a popular table that shows how well a classification model performs when applied to a set of test data with known values[85]. This can be accomplished by comparing the anticipated values of the model to the actual values. Most frequently, this table is given in matrix format. The confusion matrix itself is not difficult to comprehend; nevertheless, the language used in conjunction with it might be challenging to grasp. The application of a confusion matrix is a well-known way for analyzing and summarizing the performance of a classification system. The information recorded in a confusion matrix is utilized in the calculations that determine precision, recall, and accuracy[86].

A confusion matrix is often used to evaluate a classifier in classification analysis. In the Table (2.1), the columns show the predictions made by the classifier, while the rows show the actual classes. (True

Positive) which is what TP stands for, is the number of cases that have been correctly labeled as positive. The FN is the number of positive cases that were recorded as negatives by mistake (False Negative). TN (True Negative) is the number of negative cases that are correctly labeled as negative. FP (False Positive) is the number of negative cases that are mistakenly labeled as positive (True Negative).

Table 2.1 confusion matrix

Actual	**Classified**	
	Positive	**Negative**
Positive	TP	TN
Negative	FP	FN

When modeling unbalanced data, the class that is underrepresented is often called the "positive class," and the class that is overrepresented is often called the "negative class." Most of the performance metrics used in classification tasks are built on top of the confusion matrix. Some of these performance metrics are listed in the Table (2.2) below.

Table 2.2 Show some of the common performance metrics that are dependent on the confusion matrix

Methods	**Equation**
Recall	$\mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$
Precision	$\mathrm{TP}/(\mathrm{TP}+\mathrm{FP})$
F1-Sscore	$2\times((\mathrm{Precision}\times\mathrm{Recall})/(\mathrm{Precision}+\mathrm{Recall}))$

Accuracy	$((\mathrm{TP}+\mathrm{TN})/(\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}))$
Brier score loss	$1-\mathrm{Accuracy}$
True Positive Rate (TPR)	$\mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$
True Negative Rate (TNR)	$\mathrm{TN}/(\mathrm{TN}+\mathrm{FP})$
False Positive Rate (FPR)	$\mathrm{FP}/(\mathrm{FP}+\mathrm{TN})$
False Negative Rate (FNR)	$\mathrm{FN}/(\mathrm{FN}+\mathrm{TP})$

Accuracy is the most frequently discussed feature of classifiers in reports. For an algorithm, this statistic indicates how well it works in practice. However, as previously shown, expected accuracy can be a misleading evaluation statistic when the data are uneven. This is because, under these conditions, the majority class is given greater weight than the minority class, making it difficult for a classifier to do well in the minority class. Positive cases are evaluated by sensitivity and negative ones by specificity, respectively. There are two types of performance metrics: sensitivity and specificity. It is the classifier's sensitivity that determines how effective it is, while its specificity determines how effective it is when applied to the negative/majority group. The sensitivity and specificity of an analysis are almost always inversely proportional. A model's accuracy, on the other hand, refers to how well it captures reality. One approach to tell if a classifier is accurate is to look at its precision value, which will be higher than that of other classifiers.

There should be a balance between false positive and false negative results for analysts in general. The metrics that have been provided thus far do not provide an appropriate evaluation basis in this

situation. False positives and false negatives are equally weighted in the following performance metrics:

2.8.1 Mean Absolute Error

The mean absolute error (also known as MAE) is yet another quality measurement that finds common use in the field of autonomously generated continuous functions (models). It is defined as the average absolute difference between the value that was anticipated and the value that was really attained. Specifically, it compares the value that was predicted A_i^{Δ} to the value that was actually obtained A_i (or right)[85]:

$$MEA = \frac{1}{n}\sum_{i=1}^{n} |A_i - A_i^{\Delta}| \qquad (2.6)$$

Where: n=number of errors, A_i= actually value, A_i^{Δ}= predicted value.

2.8.2 Root Mean Squared Error

The standard deviation of the residuals is equivalent to the root mean square error, or RMSE (prediction errors). This residuals' dispersion is measured using a statistic called the root-mean-square error (RMSE). To put it another way, it shows how tightly the data is clustered around the line of best fit. The use of root mean square error as a technique for experimental data validation is common in the domain of regression analysis[87].to calculate RMSE value by using equation (2.7):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(A_i - A_i^{\Delta}\right)^2} \qquad (2.7)$$

2.8.3 Geometric Mean

The root of the equation that calculates class-wise sensitivity is the geometric mean, sometimes referred to as the G-mean. This metric aims to balance the accuracy levels of each class while maximizing their overall accuracy. The G-mean is calculated for binary classification by taking the

square root of the product of sensitivity and specificity[88]. In circumstances involving many classes, the solution is the higher root of the product of each class's sensitivity. Even if negative examples are accurately labeled as such, a low G-Mean indicates poor performance in classifying positive occurrences. Even if the examples were appropriately labeled as negative, this holds true. This care is necessary to avoid overfitting the negative class and underfitting the positive class, respectively. to calculate G-mean value by using equation (2.8):

$$G - mean = \sqrt{specificity * \ sensitivity} \qquad (2.8)$$

2.8.4 Matthews correlation coefficient

A more reliable statistical measure is the Matthews correlation coefficient (MCC)[89], It only results in a high score being given if the forecast delivers strong results in all of the confusion matrix categories, according to the amount of both positive and negative items contained within the dataset. Because of this, there is no doubt that the MCC is a reliable way to measure correlation[90]. Equation (2.9) for calculation MCC:

$$MCC = \frac{(TP*TN)-(FP*FN)}{\sqrt{(TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)}} \qquad (2.9)$$

2.8.5 Cohen's Kappa

The initial purpose of Cohen's Kappa was to assess how closely two observers agreed on how to rank the same group of people on a nominal scale with two or more classes. The metric is also frequently used for two-class categorization issues. use the individual cells that comprise a standard confusion matrix[91]. The following is the definition of Cohen's Kappa Equation (2.10):

$$CK = \frac{2*((TP*TN)-(FP*FN))}{(TP+FP)*(FP+TN)+(TP+FN)*(TN+FN)} \quad (2.10)$$

Chapter Three
System and Methodology

3.1 Introduction

In this chapter, introduce the design of a system to detect distributed denial of service attacks on Internet of Things networks. The IoT network consists of Raspberry Pi and IoT sensors. The proposed system goes through three stages. The first stage is the process of calculating the randomness of the grid using entropy every 2 seconds. The second stage involves passing a sliding time window to an early detection method using entropy threshold value. In the third stage, the decision is made whether the packet is attacked or not using machine learning algorithms (KNN and SVM) trained using the IoT-Botnet UNSW-2018 and CIC-DDoS Benchmark dataset.

3.2 The Proposed System's Basic Procedures

Our proposed system is implemented in real-time using IoT sensors that collect data and send it to the fog layer. The fog layer performs the processing and analysis of packets in order to ensure the integrity of the network using machine learning algorithms as shown in figure (3-1). The general steps for implementing our proposed system consist of:

1. IoT layer:

   It includes IoT devices and the Raspberry Pi. The sensors sense the surroundings and send the data using the Raspberry Pi to the fog layer using (TCP, UDP).

2. fog layer:

   In this layer, perform early and robust classification of network data using machine learning algorithms and Entropy.

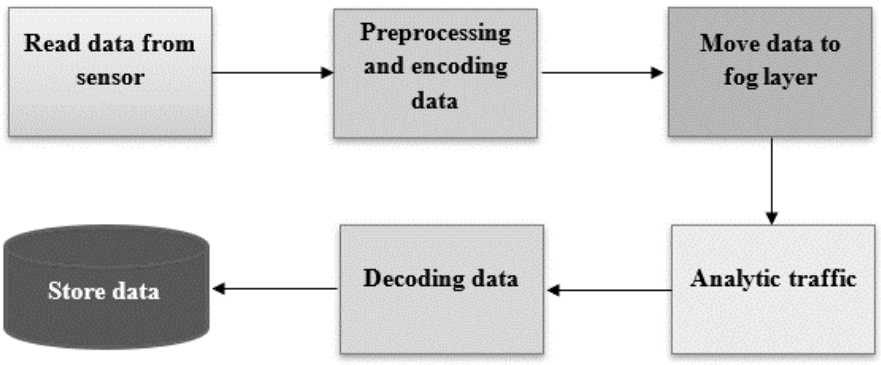

Figure 3.1 General step of propose system

3.3 Propose system for DDoS detection

The work technique is divided into two layers: the first is IoT devices layer, and the second is the fog layer. The IoT devices layer is the first layer. As shown in the Figure (3.2).

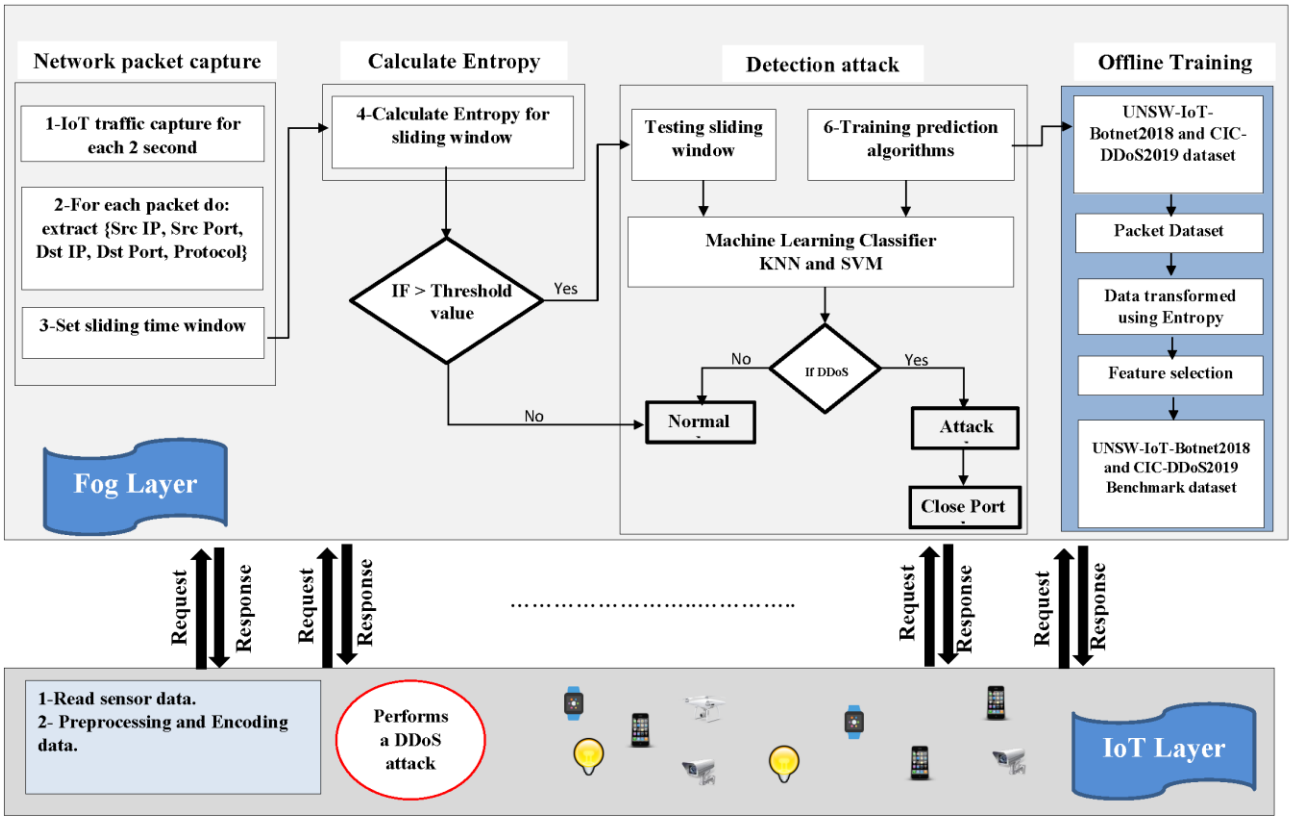

Figure 3.2 The propose system for DDoS detection

3.3.1 Network packet capture

Wireshark is used in conjunction with Python and the pyshark library to gather packets every 2 seconds. The five attributes of each future package (Source IP, Source port, Destination IP, Destination port and Protocol) are extracted and entered into the sliding window time.

Algorithm (3.1): Network packet capture

Input network capture
Output sliding time window
Begin
While every income packet **do**
Filter packet of null:
Pass;
Extract:
(Source IP/port, Destination IP/port, Protocol);
Added to sliding window;
End

3.3.2 Calculate the entropy

determine the likelihood of each IP and port via mathematical labor, and do it using both the numpy and collections libraries, to ensure that get the most accurate results. Between now and then, after computing the probability, compute the randomness for each of the four attributes that have extracted using Shannon entropy. These data points are sent on to the prediction algorithms for analysis.

Algorithm (3.2): Calculate the entropy

Input sliding time window from real-time
Output four point Entropy
For each sliding window **do**
Begin
C = Collection counter for (IP,port);
Counts = Count number of C value;
P = Count/sumCount;
H = sum (-(P*log(P)));
End

3.3.3 Detection and mitigation attack

At this stage, use machine learning algorithms. Use two types of algorithms (SVM and KNN) as well as use entropy to set a threshold value, which will be mentioned in detail.

A dynamic suggested system depicted in figure (3.3) is utilized proactively to identify abnormalities such as DDoS assaults. Then, these taught patterns (signatures) are evaluated using data from the testing set.

For early DDoS attack detection, applied the entropy threshold value. This approach enables a security analyst to monitor and identify huge floods of DDoS assaults in an efficient period. The patterns generated in the training step is utilized for categorizing the unknown packets from the network flow. The datasets comprise of massive network traffic from IoT-Botnet UNSW-2018 assaults and CIC-DDoS 2019 dataset.

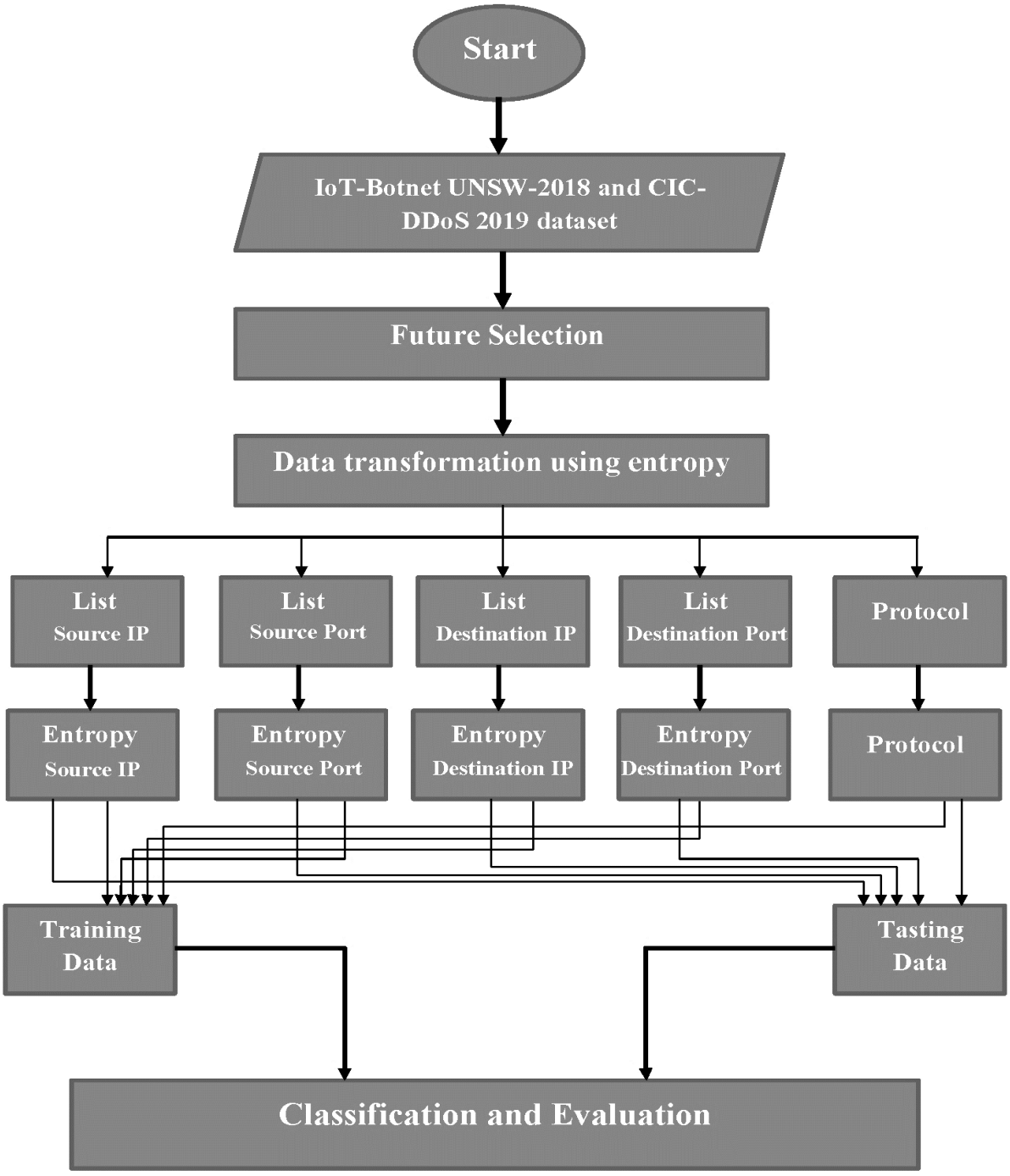

Figure 3.3 Training and testing on offline dataset to building Benchmark dataset.

3.3.3.1 Early detection

The idea is to use entropy to set a threshold value for the Source IP address. If the real-time entropy calculation of the Source IP address

goes above the threshold, an alert is sent to the admin about DDoS attacks, and suspicious traffic is sent to machine learning algorithms to see if an attack has already happened.

Algorithm 3.3: Entropy to set a threshold value

Input (**sliding window$_{(P)}$** = Entropy for (Source IP))
Output (Attack / Normal)
Begin

If sliding window$_{(P)}$ > threshold value **do**

alert the admin about DDoS attacks;

send sliding window to ML algorithms for the final diction

End

3.3.3.2 KNN (K-Nearest Neighbors algorithm)

the data will be named based on their proximity and distance from attack points and natural points that are categorized in advance in the datasets. In this research use Euclidean Distance because it is not affected by dimensions, all dimensions will be calculated correctly[92].

Algorithm (3.4): Pseudo code of a KNN algorithm

Input (Entropy for (Src_IP, Src_port, Des_IP, Des_port))
Output (Attack / Normal)
1- **Begin**
2- Load the train data.
3- Set the value of K = 3.
4- Calculate the distance between the testing sliding window from real time with all rows of the training set.
   4.1. Sort the distances that have been calculated in ascending order based on the distance value.
   4.2. Choose the first K rows from the sorted array.
   4.3. Find the class that appears in label data.
   4.4. Return the projected class label.
5- **End.**

The Algorithm (3.4) shows the KNN algorithm was built and trained using the IoT-Botnet UNSW-2018 and CIC-DDoS2019

Benchmark dataset. The algorithm chooses the three closest neighbors to the predicted point from the training data, after which the projected class label is returned.

3.3.3.3 Support Vector Machine (SVM)

For an SVM to generalize across two separate classes, it needs a collection of labelled data as input in order for the algorithm to train on. The SVM's primary job is to look for a hyperplane that can tell the difference between two classes.

Several hyperplanes can accomplish this duty, but the goal is to identify one that allows for the greatest distances between the two classes, so that any additional data points that need to be classed can be done quickly and accurately in the future. rbf is one of the most commonly utilized kernels (Radial Basis Function)[79].

Algorithm (3.5): Pseudo Code for SVM algorithm

Input (**sliding window** =Entropy for **(Src_IP, Src_port, Des_IP, Des_port)**)
Output (Attack / Normal)
Begin
Training data:
1. Load the train data (**IoT-Botnet UNSW-2018 and CIC-DDoS**)
2. Apply kernel function for SVM (rbf).
3. Specify hyperplane**.**

Test data
1. Put **sliding window** to predict it class label
2. Label **sliding window**

End

The Algorithm (3.5) shows the SVM algorithm was trained on IoT-Botnet UNSW-2018 and CIC-DDoS2019 using the kernel function (rbf) to select hyperplane to ensure easy and fast prediction.

3.3.3.4 Close port

In addition, the port where the attack occurred is closed if the algorithm's rating and entropy indicate an attack as shown as algorithm (3.6).

Algorithm 3.6 close or open port class

Input (Port)
Output (Close Port)

Start

For close port Do:
Sudo ufw deny Port

End

Algorithm (3.7) Overview of the proposed system

Input network capture
Output Attack or Normal
Begin
While every income packet **do**
Filter packet of null:
Pass;
Extract:
(Source IP/port, Destination IP/port, Protocol);
Added to sliding window;
Calculate the entropy for sliding window
C = Collection counter for (IP,port);
Count = Count number of C value;
P = Count/sumCount;
H = sum (-(P*$\log_2$(P)));

sliding window **(Point)**=Entropy for (Src_IP, Src_port, Des_IP, Des_port))
Entropy threshold value

If Entropy for (Source IP) > threshold value **do**
alert the admin about DDoS attacks;
send **sliding window** **(Point)** to machine learning algorithms for the final diction
In the same time run (KNN and SVM) class

Load the train data (IoT-Botnet UNSW-2018 and CIC-DDoS2019**)** for both KNN and SVM

Set the value of **K = 3** and calculate the distance for KNN algorithm.

Apply kernel (rbf) and Specify hyperplane for SVM

If any algorithms agree with entropy threshold value **do**:

Close port

Label **sliding window** as attack

Else

Label **sliding window** as normal

End

Chapter Four Intrusion Detection System's (IDS) Result

4.1 Introduction

This chapter details the implementation of the suggested system as well as the outcomes of its use. The proposed system is put through its paces using a variety of malicious datasets, including the CICDDoS2019 and UNSW IoT Botnet 2018 datasets, among others. The DDoS attacks that are a part of CICDoS2019 are designed to be safe and up to date, and they are meant to be as accurate a simulation of real-world data as possible. The University of New South Wales in Canberra (UNSW Canberra), made it possible to create the BoT-IoT dataset. The environment of the network contained both regular internet traffic and traffic from botnets, in addition to other types of internet traffic. These datasets explanations are provided in the subsequent subsections of this section.

4.2 Requirements for System Installation

The proposed system is characterized by the ease of installation in the IoT networks and provides protection in the event that the network is exposed to distributed denial of service attacks. Our proposed system consists of:

1. Internet of things devices (temperature sensor, and temperature-humidity sensor).
2. Raspberry Pi (contains the Raspbian operating system).
3. Computer system (running on the Ubuntu 22.04 operating system).

4.3 Particulars about the device node

The IoT layer has two Raspberry Pi 400 with 4GB of RAM and 32 GB MicroSD card, a temperature sensor, and a temperature and humidity sensor as shown in Table 4.1.

Table 4.1 details about node

Device	Model	Number	Size	Range	Voltage
Raspberry Pi	400	2	12*28 cm	-	5v
Temperature Sensor	DS18B20 Waterproof Digital	2	6*50mm	-55 to +125 °C	3 to 5.5 V
Temperature and humidity sensor	DHT11 Digital	2	3cm*1cm	Temperature: 0°C to 50°C Humidity: 20% to 90%	3.5V to 5.5V

4.4 Details Regarding the Characteristics of Network Traffic

There were one million packets from each type of DDoS assault in the CICDDoS2019 and UNSW IoT Botnet 2018 datasets that were selected at random and saved in an associated file. Then, a computer language was used to read from the saved file in order to analyze the data.

Only the TCP/IP ensemble's header information is used by proactive systems based on extracted ensemble pattern data. The majority of the discovered attributes (characteristics) are provided in the Table(4.2) since the payload has been eliminated from the CICDDoS2019 and UNSW IoT Botnet 2018 datasets.

Table 4.2 Selection feature in CICDDoS2019 and UNSW IoT Botnet 2018 datasets

No.	Feature	Layer	Information	Dataset
1	saddr	Network Layer	Source IP address	CICDDoS2019 and UNSW IoT Botnet 2018
2	daddr	Network Layer	Destination IP address	
3	sport	Transport Layer	Source Port	

4	dport	Transport Layer	Destination Port	
5	Protocol	TCP/IP suite	TCP, UDP, HTTP and etc.	

CICDoS2019 contains the most current instances of distributed denial of service (DDoS) attacks. Only their distributed denial of service attack, which targets DNS, LDAP, MSSQL, Net BIOS, NTP, PORT MAP, SNMP, SSDP, TFTP, UDP, and UDP Lag, could be extracted. Again, were only able to extract TCP, UDP, and HTTP DDoS attack related data samples from the UNSW IoT Botnet 2018 dataset, despite it including three unique types of attacks. The total number of datasets that were categorized based on whether they contained benign or malicious DDoS payloads is displayed in Table (4.3).

Table 4.3 Explained total number of records

Dataset	Protocol attack	Total number of records
CICDoS2019	DNS	1046500
	LDAP	1047795
	MSSQL	1047561
	Net BIOS	1047700
	NTP	1035033
	PORT MAP	186976
	SSDP	1048590
	TFTP	1048576

	UDP	1047441
	UDP Lag	366488
	SYN	1048264
	Normal data	29533
Bot-IoT2018	TCP	1048577
	UDP	948289
	HTTP	1000000
	Normal data	25733

4.5 Preprocessing data

The collection of data consists of one file per attack class. There are twelve attack classes in CICDDOS2019, compared to three in UNSW IoT Botnet 2018. The distinguishing characteristic of each attack class consists of diverse network conditions. To verify the proposed system's ability to accurately detect each attack, each type of attack's data is processed individually. After evaluating exploratory data and resolving outliers and missing values, the data is then transformed for training and testing.

4.5.1 Data transformation using entropy

Table (4.4) shows the number of packets captured in real time in 2 second when the network is in a steady state.

Table 4.4 show number of packets captured in 2 second

Source IP	Source port	Destination IP	Destination port	Protocol
192.168.0.9	49354	192.168.0.100	9090	TCP
192.168.0.100	9090	192.168.0.9	49354	TCP
192.168.0.100	9090	192.168.0.9	49354	TCP
192.168.0.9	49354	192.168.0.100	9090	TCP
192.168.0.9	49354	192.168.0.100	9090	TCP
192.168.0.9	49356	192.168.0.100	9090	TCP
192.168.0.9	49354	192.168.0.100	9090	TCP
192.168.0.100	9090	192.168.0.9	49356	TCP
192.168.0.9	49356	192.168.0.100	9090	TCP
192.168.0.10	42624	192.168.0.100	9090	TCP
192.168.0.100	9090	192.168.0.10	42624	TCP
192.168.0.100	9090	192.168.0.10	42624	TCP
192.168.0.10	42624	192.168.0.100	9090	TCP
192.168.0.10	42624	192.168.0.100	9090	TCP
192.168.0.10	42626	192.168.0.100	9090	TCP
192.168.0.9	49354	192.168.0.100	9090	TCP
192.168.0.100	9090	192.168.0.10	42624	TCP
192.168.0.100	9090	192.168.0.10	42626	TCP
192.168.0.100	9090	192.168.0.9	49354	TCP
192.168.0.10	42626	192.168.0.100	9090	TCP
192.168.0.9	5900	192.168.0.100	57287	TCP
192.168.0.100	57287	192.168.0.9	5900	TCP
192.168.0.100	57287	192.168.0.9	5900	TCP
192.168.0.9	5900	192.168.0.100	57287	TCP
192.168.0.9	5900	192.168.0.100	57287	TCP
192.168.0.100	57287	192.168.0.9	5900	TCP
192.168.0.100	56865	149.154.175.100	443	TCP
192.168.0.100	56866	149.154.175.100	80	TCP
192.168.0.9	49356	192.168.0.100	9090	TCP
192.168.0.9	49360	192.168.0.100	9090	TCP
192.168.0.100	9090	192.168.0.9	49356	TCP
192.168.0.100	9090	192.168.0.9	49356	TCP

Depending on the Table (4.4) give an example showing how to calculate the entropy of packets during real time using both collections and numpy libraries:

Source IP:

Calculate the entropy of the first column, Source IP, using the collections library that defines each IP's how many iterations are in the Source IP list.

Collections of Source IP= {'192.168.0.100': 15, '192.168.0.9': 12, '192.168.0.10': 5}

Use counts to extract the number of iterations for each Source IP.

counts of Source IP= [12. ,15. ,5.]

Calculate the probability of a source IP by dividing the counts by the sum of the counts.

The probability of 192.168.0.100 = 15/32 = 0.375

The probability of 192.168.0.9 = 12/32 = 0.46875

The probability of 192.168.0.10 = 5/32 = 0.15625

Calculate entropy:

$$H(\propto) = -(\frac{15}{32}\log_2\frac{15}{32} + \frac{12}{32}\log_2\frac{12}{32} + \frac{5}{32}\log_2\frac{5}{32})$$

$$H(\propto) = -((-0.512395) + (-0.530639) + (-0.418448))$$

$$H(\propto) = -(-1.461482831)$$

$$H(\propto) = 1.461482831$$

Destination IP

Collections of Destination Port= {'192.168.0.100': 17, '192.168.0.9': 9, '192.168.0.10': 4, '149.154.175.100': 2}

counts of Destination port= [17. 9. 4. 2.]

The probability of 192.168.0.100 = 17/32 = 0.53125

The probability of 192.168.0.9 = 9/32 = 0.28125

The probability of 192.168.0.10 = 4/32 = 0.125

The probability of 149.154.175.100= 2/32 = 0.0625

$$H(\propto) = -(\frac{17}{32}\log_2\frac{17}{32} + \frac{9}{32}\log_2\frac{9}{32} + \frac{4}{32}\log_2\frac{4}{32} + \frac{2}{32}\log_2\frac{2}{32})$$

$$H(\propto) = -((-0.484785) + (-0.514708) + (-0.375) + (-0.25))$$

$$H(\propto) = -(-1.624493)$$

$$H(\propto) = 1.624493$$

Using the same strategy, the sliding window idea is also applied to the CICDDOS2019 and UNSW IoT Botnet 2018 datasets. Entropy is utilized to quantify the quantity of this unpredictable event. The idea of entropy reduction is an excellent signal for identifying assaults such as DDoS attacks, in which the entropy values of the source IP addresses will grow and the entropy values of the destination IP addresses will decrease throughout the attack period. Table (4.5) displays the final data count for both data sets after the Entropy transformation procedure has been executed.

Table 4.5 Show final data count for CICDDOS2019 and UNSW IoT Botnet 2018 datasets

Dataset	Protocol attack	Entropy transformation
CICDoS2019	DNS	20930
	LDAP	20956

	MSSQL	20952
	Net BIOS	20954
	NTP	20700
	PORT MAP	3739
	SSDP	20971
	TFTP	20971
	UDP	20948
	UDP Lag	7329
	SYN	20964
	Normal data	590
Bot-IoT2018	TCP	20971
	UDP	18965
	HTTP	20000
	Normal data	260

4.5.2 Feature Selection

Feature selection is the process of manually or mechanically choosing which features will be included in a product. It is conceivable that the existence of insufficient features in the data will result in a deterioration in the quality of the models and drive them to train based on less relevant attributes. The range of viable options:

4.5.2.1 Chi-square

It is a filter strategy for evaluating features using statistical methods, and it reduces the chance that the model is overfit. If the value of chi-square is high, there is a larger dependence of features on response; if it is low, there is less dependence.

In the proposed DDoS detection system, the chi-square method is used to analyze the five proposed properties for syn attack that are chosen from the CICDoS2019 dataset, which are as follows: Saddr, Sport, Daddr, Dport, the protocol, and the expectation. The results of this analysis are displayed in the table below:

Table 4.6 chi-square value

Feature	Src_ip	Src_port	Des_ip	Des_port	Protocol
chi-square value	31692	75	60593	1148	0
Probability value	0.0	0.0	0.0	0.0	1.0

The value of the chi-square statistic is displayed in the first row of Table (4.6), while the value of the probability is displayed in the second row. According to the data presented in the table above, src_ip, src_port, des_ip, and des_port all received a high value. This means that there is a greater dependency on the characteristics, whereas the value 0 that Protocol received shows that it is independent and has the lowest value.

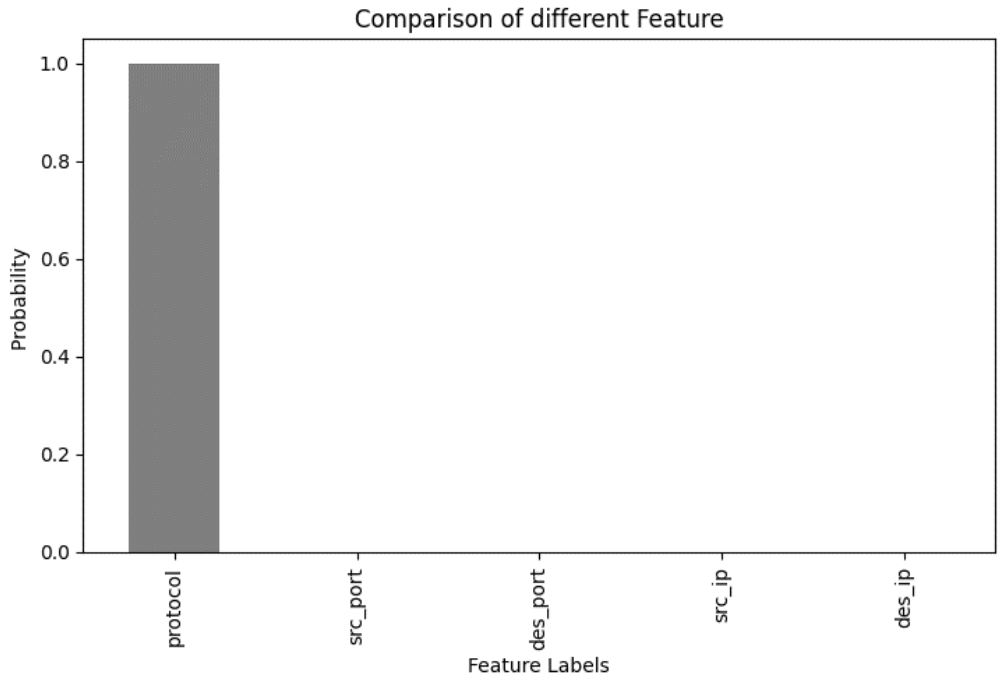

Figure 4.1 Show p-value for the five features

Figure (4.1) shows that since the p-value for the Protocol feature is higher, it means that the variables in question have nothing to do with the outcome and can't be taken into account when training.

4.5.2.2 Extra Tree feature selection

It is an ensemble method that randomly produces numerous tree models from the training dataset and then sorts the most well-liked features. An illustration of this kind of algorithm is the Extra Tree Classifier. In this model-based approach to feature selection, tree-based supervised models are used to help assess the relative value of the various attributes. When fitting each decision tree, it uses the entire dataset rather than a bootstrap replica and chooses a random split point to divide the tree's nodes. The Extra Tree Classifier parameters are shown in Table (4.7).

Table 4.7 Extra Tree Classifier parameters and setting

Parameter	**n_estimators**	**criterion**	**max_features**
Setting	5	Entropy	2

Explanation of the columns in Table (4.7) n estimators equals 5. This indicates that five decision trees are utilized during execution to select the best property. criterion = "entropy" is a function for measuring the split's quality by use Information Gain; max features = 2 This means that each decision tree will test two characteristics.

Figure (4.2) depicts the final output of the Extra Tree Classifier method when applied to the five features for TCP attack that are chosen from the Bot-IoT2018 dataset.

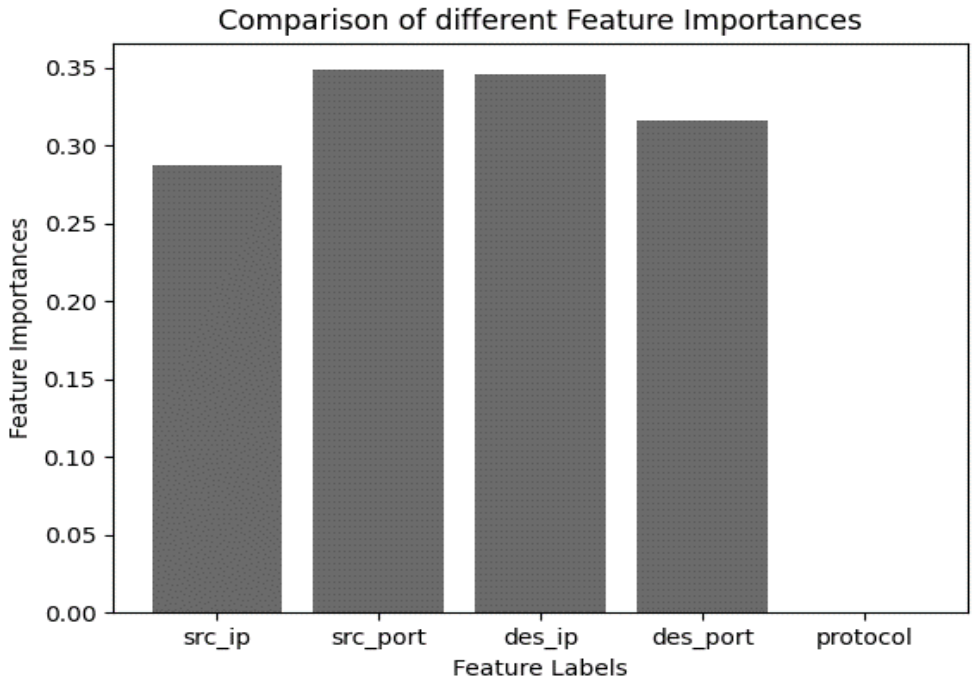

Figure 4.2 Feature selection Extra Tree

According to Figure (4.2), the feature "src_port" is the most essential variable to consider when determining the output label based on

Extra Trees, whereas the feature "protocol" is the variable that should be given the least amount of consideration.

In light of the fact that all of the classifiers have reached a consensus regarding the selection of the most important characteristics, the "protocol" feature ought to be disregarded when training the data in order to ensure that both training and testing are confined to a maximum of only four characteristics.

4.5.3 Machine learning algorithms Analysis and Results

The suggested method is evaluated in three steps for DDoS detection: The basic stage entails testing it against the CICDDoS2019 database. The second stage of the Bot-IoT2018 database has been reached. The third and last stage is testing the system in real-world conditions. Using KNN and SVM algorithmic techniques in the suggested system, the parameters to be employed by each classifier are provided below Table 4.8): -

Table 4.8 Component and tuning for KNN and SVM algorithms

Algorithms	Components	Tuning
KNN	n_neighbors	3
	weights	uniform
	algorithm	auto
	leaf_sizeint	30
	p	2 (Euclidean distance)
	metrics	minkowski

SVM	C	1.0
	kernel	rbf
	Degree	3
	Gamma	scale

4.5.3.1 Evaluation on CICDDoS2019

This database contains various distributed denial of service attack types. The proposed system was evaluated on 7 types of attacks, with a 30% test and 70% training for each.

As shown in Table (4.9), **the syn attack** is the first type of attack that the proposed system has been tested against.

Table 4.9 Evaluation propose system on syn attack

Evaluation metrics	KNN	SVM	Evaluation metric	KNN	SVM
Time in second	1.21	0.60	FPR	0	0
Recall	1.00	1.00	FNR	0	0
Precision	1.00	1.00	MAE	0	0
F1-Sscore	1.00	1.00	RMSE	0	0
Accuracy	1.00	1.00	G-Mean	1.00	1.00
Brier score loss	0	0	MCC	1.00	1.00
TPR	1.00	1.00	CK	1.00	1.00
TNR	1.00	1.00	Log_loss	0	0

Table (4.9) demonstrates that the process of detecting flood attacks using the KNN and SVM algorithms was highly accurate, with a difference in the detection times for each algorithm. The KNN algorithm required 1.26 seconds to be discovered, while the SVM algorithm required only 0.60 seconds.

On the proposed system, the **UDP attack** selected from the CICDDoS2019 database is evaluated as shown in Table (4.10).

Table 4.10 Evaluation propose system on UDP attack

Evaluation metrics	KNN	SVM	Evaluation metric	KNN	SVM
Time in second	1.81	0.60	FPR	0	0.0015
Recall	1.00	100	FNR	0	0.00031
Precision	1.00	100	MAE	0	0.00045
F1-Sscore	1.00	100	RMSE	0	0.017
Accuracy	1.00	0.9995	G-Mean	1.00	0.9971
Brier score loss	0	0.0005	MCC	1.00	0.9916
TPR	1.00	0.9996	CK	1.00	0.9916
TNR	1.00	0.9945	Log_loss	0	0.016

Using KNN and SVM algorithms, the proposed system achieved high accuracy in detecting a UDP attack. KNN is rated highly for its ability to handle non-separable data, while SVM is rated at 99.95. This is probably the reason for the non-separable data as shown in Figure (4.3).

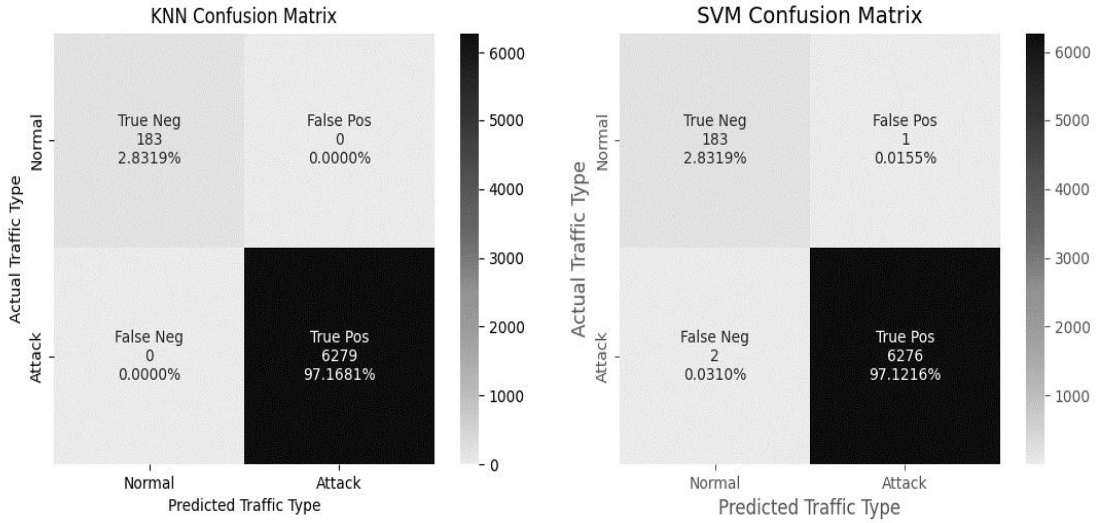

Figure 4.3 Confusion matrix for UDP attack (KNN and SVM)

In terms of the MAE scale, discovered that there was a disparity of 0.045 between the expected value and the obtained value as shown in Table (4.10).

Port-map attack, which is used to search for open ports on a victim's device, is one of the initial methods of attack that an attacker can utilize. The proposed defense mechanism was put to use in order to thwart this assault, and the outcomes are detailed in Table (4.11).

Table 4.11 Evaluation propose system on Port-map attack

Evaluation metrics	KNN	SVM	Evaluation metric	KNN	SVM
Time in second	0.31	0.22	FPR	0	0
Recall	1.00	1.00	FNR	0	0
Precision	1.00	1.00	MAE	0	0
F1-Sscore	1.00	1.00	RMSE	0	0
Accuracy	1.00	1.00	G-Mean	1.00	1.00

Brier score loss	0	0	MCC	1.00	1.00
TPR	1.00	1.00	CK	1.00	1.00
TNR	1.00	1.00	Log_loss	0	0

The G-Mean value in Table (4.11) suggests that 1 is correct. This reveals the degree to which the classifiers for the positive categories are effective in their categorization. A correlation coefficient between the categories that were expected and those that were seen are displayed by MCC. A correlation of 1 shows a perfect forecast, often known as an ideal value for the correlation. whereas CK achieved a value of 1, which represents a situation in which the model's prediction and the actual categories are in perfect accordance with one another.

The next type of attack chosen is the **MSSQL attack**. In order to launch distributed denial of service attacks, attackers take use of the Microsoft SQL Server Resolution Protocol. The proposed system mechanism was put to the test to identify this particular form of assault, and the results of those tests are presented in the Table below (4.12).

Table 4.12 Evaluation propose system on MSSQL attack

Evaluation metrics	KNN	SVM	Evaluation metric	KNN	SVM
Time in second	1.88	0.68	FPR	0.0153	0.01818
Recall	1.00	0.99	FNR	0	0.0003
Precision	1.00	0.99	MAE	0.00046	0.00077
F1-Sscore	1.00	0.99	RMSE	0.0215	0.0278
Accuracy	0.9995	0.9993	G-Mean	0.9922	0.9907

Brier score loss	0.0005	0.0007	MCC	0.99204	0.98441
TPR	1.00	0.9996	CK	0.99200	0.98440
TNR	0.9846	0.9818	Log_loss	0.0160	0.0267

Log loss is one of the international standards used to evaluate the performance of our suggested system. Its value varies from 0 to 1, with 0 representing the lowest value and 1 representing the maximum. This shows that the suggested system's prediction accuracy is high, as the likelihood of errors is low. Table (4.12) demonstrates that both the KNN and SVM algorithms achieved a low log-loss value (KNN=0.0160 and SVM=0.0267).

LDAP attack is the fifth attack studied against the proposed system. Attackers use the LDAP protocol to generate distributed denial-of-service attacks by flooding the server with tiny requests and bringing it to a halt in order to react to these fabricated addresses.

Table 4.13 Evaluation propose system on LDAP attack

Evaluation metrics	KNN	SVM	Evaluation metric	KNN	SVM
Time in second	1.27	0.89	FPR	0.012048	0.01754
Recall	1.00	1.0	FNR	0.000635	0.0025
Precision	1.00	1.0	MAE	0.00092	0.0029
F1-Sscore	1.00	1.0	RMSE	0.0304	0.05421
Accuracy	0.99907	0.99707	G-Mean	0.9936	0.9899
Brier score loss	0.00093	0.00293	MCC	0.98157	0.9456
TPR	0.9993	0.9974	CK	0.98155	0.9446

TNR	0.9879	0.9824	Log_loss	0.03205	0.1015

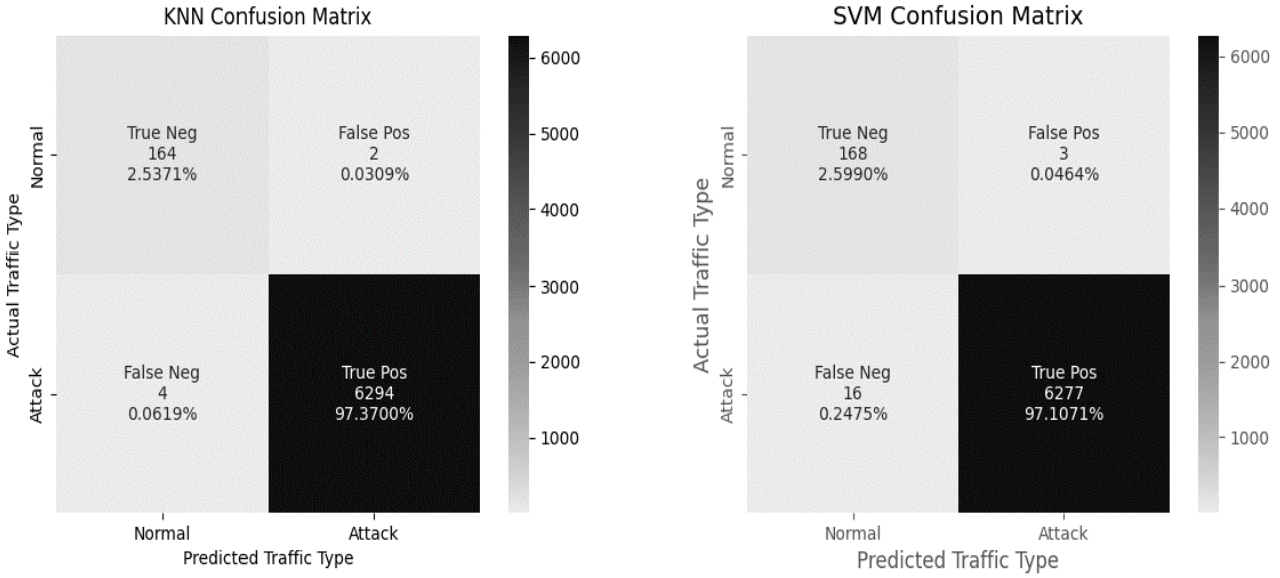

Figure 4.4 Confusion matrix for LDAP attack (KNN and SVM)

Figure (4.4) demonstrates that the number of false positives for the KNN algorithm is 2, while the number for the SVM technique is 3. This indicates that the benign traffic was identified as an LDAP assault. In addition, the number of false negatives for the KNN method is equal to 4 and the number of false negatives for the SVM algorithm is equal to 16, indicating that the LDAP assault is classed as benign traffic. As indicated in Table (4.13), these are the fundamental features of the confusion matrix that influence the typical performance metrics that depend on the confusion matrix.

Another attack used to test the proposed system is **NetBIOS attack**. Attackers send multiple inquiries utilizing the NetBIOS protocol to flood the victim to react to many fake requests as valid requests. These answers might flood and bloat the network.

Table 4.14 Evaluation propose system on NetBIOS attack

Evaluation metrics	KNN	SVM	Evaluation metric	KNN	SVM
Time in second	2.55	0.96	FPR	0.04522	0.0154
Recall	0.98	1.00	FNR	0	0.0012
Precision	1.00	0.99	MAE	0.00139	0.00170
F1-Sscore	1.00	1.00	RMSE	0.03731	0.04125
Accuracy	0.99861	0.99862	G-Mean	0.9771	0.9916
Brier score loss	0.00139	0.00178	MCC	0.9764	0.9712
TPR	1.00	0.9995	CK	0.9761	0.9711
TNR	0.9547	0.9845	Log_loss	0.0480	0.05877

According to Table (4.14), the training and detection time for both algorithms appear to have exceeded the rest of the previously used attacks. The reason for this appears to be that the data is intertwined with each other. For this attack, the SVM algorithm creates more than one hyper-plane until it reaches the optimal state of a separation between the two types of data, with some incorrect classifications, as shown in Figure (4.5). This interference also had an impact on the KNN algorithm; even if the correct positivity was categorized with high accuracy, see an inaccurate classification of the real negativity, as illustrated in Figure (4.5).

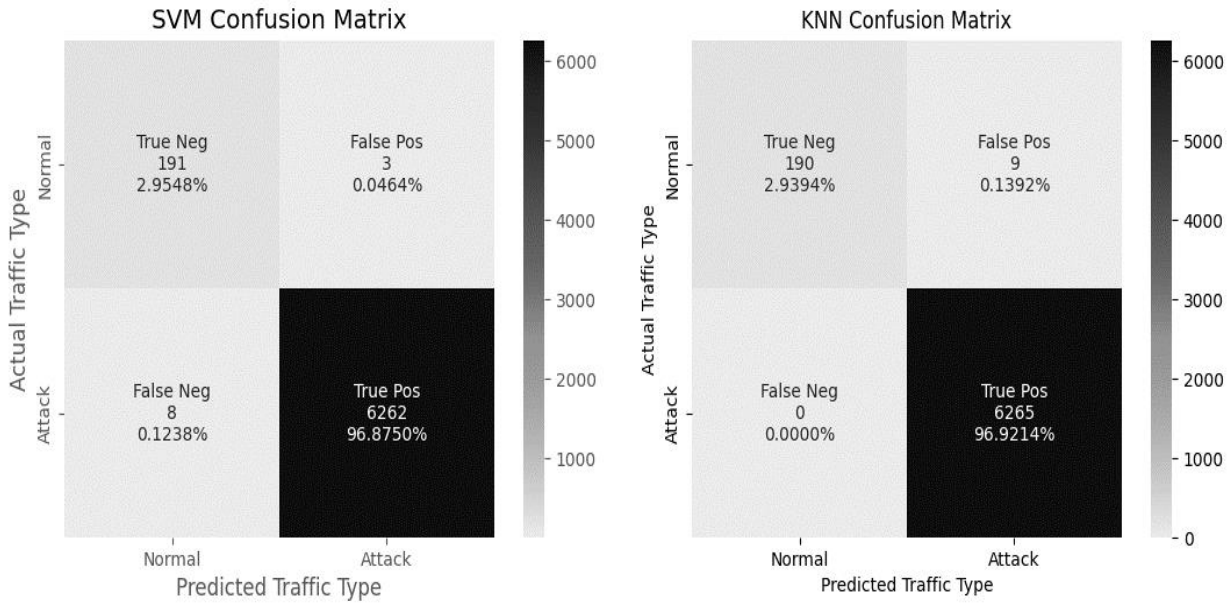

Figure 4.5 Confusion matrix for NetBIOS attack (KNN and SVM)

The UDP-Lag assault This attack attempts to absorb the bandwidth of the victim's network until it responds slowly. This vulnerability was exploited by attackers and included in online games to slow down and defeat the opponent's game. The results of testing this type of attack on the proposed system are shown in Table (4.15).

Table 4.15 Evaluation propose system on UDP_Lag attack

Evaluation metrics	KNN	SVM	Evaluation metric	KNN	SVM
Time in second	0.45	0.29	FPR	0.0062	0
Recall	1.00	1.00	FNR	0	0.00045
Precision	1.00	1.00	MAE	0.00042	0.00042
F1-Sscore	1.00	1.00	RMSE	0.02051	0.02051
Accuracy	0.9995	0.99957	G-Mean	0.9968	0.9997
Brier score loss	0.0005	0.0043	MCC	0.9966	0.99705

TPR	1.00	0.9995	CK	0.9966	0.99704
TNR	0.9937	1.00	Log_loss	0.0145	0.0145

The proposed system was also evaluated using precision, recall, and F1-score as shown in Table (4.15). Precision is a measure of algorithm quality; if the precision value is high, the algorithm displays relevant results more frequently than irrelevant ones. It is also known as a positive prediction. Recall is a measurement of quantity. The greater the recall, the more frequently the algorithm displays the most relevant results; this is known as sensitivity. F1-scores are the harmonic mean of recall and precision.

All prior attacks were compiled into a single file, and the proposed system was evaluated against various sorts of CICDDoS2019 attacks, as illustrated in Table (4.16) and Figure(4.6).

Table 4.16 Evaluation propose system on all CICDDoS attack

Evaluation metrics	**KNN**	**SVM**	**Evaluation metric**	**KNN**	**SVM**
Time in second	8.34	4.16	FPR	0.0	0
Recall	1.00	1.00	FNR	0.0081	0.0001
Precision	1.00	1.00	MAE	0.00042	0.0
F1-Sscore	1.00	1.00	RMSE	0.0089	0.0098
Accuracy	0.9999	0.9999	G-Mean	0.9999	0.9999
Brier score loss	0.0001	0.0001	MCC	0.9986	0.9983
TPR	0.9999	0.99989	CK	0.9986	0.9983

TNR	1.00	1.00	Log_loss	0.0027	0.0033

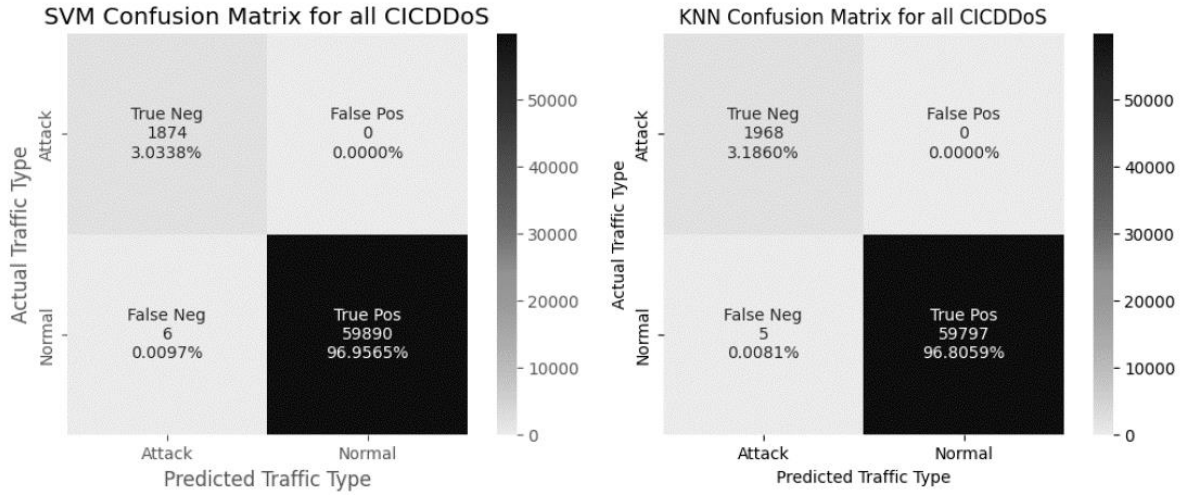

Figure 4.6 Confusion matrix for all CICDDoS2019 attack (KNN and SVM)

4.5.3.2 Evaluation on UNSW-Bot-IoT2018

UNSW-Bot-IoT2018 includes two varieties of attacks (DoS and DDoS). Only distributed denial of service (DDoS) attacks is evaluated on the proposed system. This database contains HTTP, UDP, and TCP attacks with a 25% test and 75% training for each to evaluating the proposed system.

TCP assault is the first type of attack selected from the UNSW-Bot-IoT 2018 database. This attack is used to evaluate the proposed system across 16 criteria, as given in Table (4.17).

Table 4.17 Evaluation propose system on TCP attack

Evaluation metrics	KNN	SVM	Evaluation metric	KNN	SVM
Time in second	2.32	0.85	FPR	0	0
Recall	1.00	1.00	FNR	0	0.00019
Precision	1.00	1.00	MAE	0	0.00018

F1-Sscore	1.00	1.00	RMSE	0	0.01364
Accuracy	1.00	0.9998	G-Mean	1.00	0.9999
Brier score loss	0	0.0002	MCC	1.00	0.9960
TPR	1.00	0.9998	CK	1.00	0.9960
TNR	1.00	1.00	Log_loss	0	0.0064

In Table (4.17), the KNN algorithm classified the attack with 100% accuracy and a Log loss ratio of zero, indicating that the classification was flawless. In contrast, the SVM algorithm achieved a 99% accuracy due to the lower percentage of true positivity than the KNN algorithm. In terms of time, the SVM method was three times faster than the KNN approach to detecting attacks. In general, the suggested system utilizing both methods detect this type of assault with excellent precision.

As indicated in Table (4.18), the suggested system was evaluated using the second type of attack from the UNSW-Bot-IoT 2018 database, a **UDP assault**.

Table 4.18 Evaluation propose system on UDP attack

Evaluation metrics	KNN	SVM	Evaluation metric	KNN	SVM
Time in second	1.74	0.57	FPR	0	0
Recall	1.00	1.00	FNR	0	0
Precision	1.00	1.00	MAE	0	0
F1-Sscore	1.00	1.00	RMSE	0	0
Accuracy	1.00	1.00	G-Mean	1.00	1.00

Brier score loss	0	0	MCC	1.00	1.00
TPR	1.00	1.00	CK	1.00	1.00
TNR	1.00	1.00	Log_loss	0	0

The G-Mean Standard, which received the highest rating value, accurately identified all positive categories as shown in Table (4.18), achieving the highest rating value. In corroboration of the preceding, the CK criterion revealed that the model predictions and actual categories were in perfect agreement. Figure (4.7) displays the confusion matrix for both algorithms to validate the detection accuracy of the model.

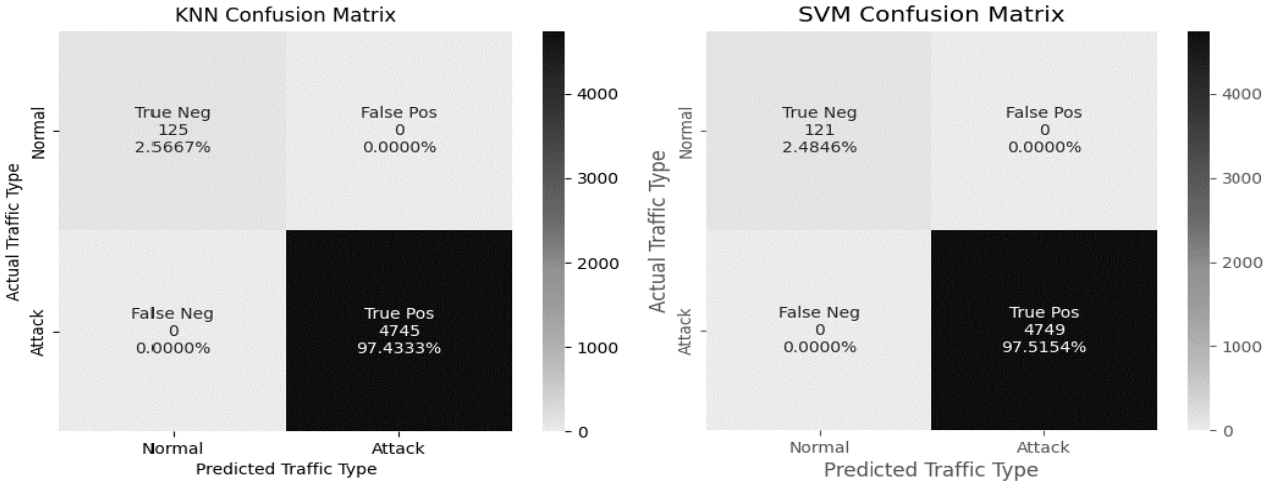

Figure 4.7 Confusion matrix for UDP attack (KNN and SVM)

The HTTP assault is one of the attacks that target the seventh layer, and it is considered one of the most current attacks because it uses less bandwidth to attack than prior attacks. This attack demands comprehensive knowledge of the victim's website or server. This attack has been evaluated against the suggested system.

Table 4.19 Evaluation propose system on HTTP attack

Evaluation metrics	**KNN**	**SVM**	**Evaluation metric**	**KNN**	**SVM**

Time in second	3.39	0.76	FPR	0	0
Recall	1.00	0.99	FNR	0	0.00119
Precision	1.00	0.99	MAE	0	0.00116
F1-Sscore	1.00	1.00	RMSE	0	0.0342
Accuracy	1.00	0.9988	G-Mean	1.00	0.9994
Brier score loss	0	0.0002	MCC	1.00	0.9751
TPR	1.00	0.9988	CK	1.00	0.9748
TNR	1.00	1.00	Log_loss	0	0.04040

The results of the proposed system evaluation against HTTP attacks are presented in Table (4.19). The KNN algorithm classifies the HTTP attack with accuracy = 100% because the number of true positives = 5009. This means that it correctly classifies all DDoS attacks because the number of false negatives is 0, the number of true negatives is 120, and it also classifies them correctly because the false positives are 0. As for the second classifier, SVM, it also got a high accuracy of 99.88%. It is noted that it is less than the previous one because it classified the number of true positives as equal to 5004, false negatives as 0, the number of true negatives as 119, and the number of false positives as 6. This shows the SVM classifier's failure to classify six points, as shown in Figure (4.8).

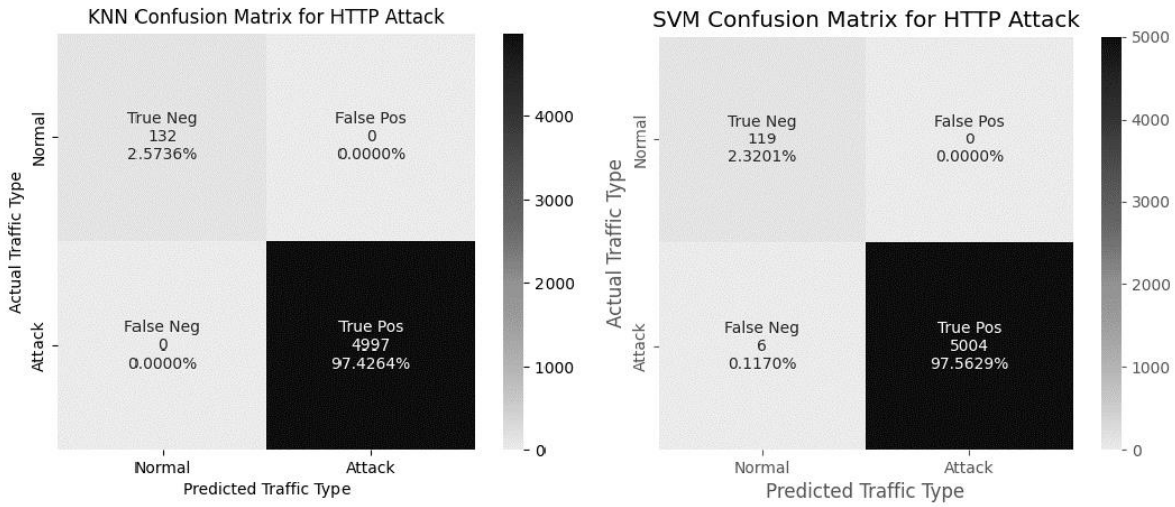

Figure 4.8 Confusion matrix for HTTP attack (KNN and SVM)

All prior attacks were compiled into a single file, and the proposed system was evaluated against various sorts of IoTBotNet2018 attacks, as illustrated in Table (4.20) and Figure (4.9).

Table 4.20 Evaluation propose system on all UNSW-Bot-IoT 2018 attack

Evaluation metrics	**KNN**	**SVM**	**Evaluation metric**	**KNN**	**SVM**
Time in second	5.19	2.65	FPR	0	0
Recall	1.00	1.00	FNR	0	0
Precision	1.00	1.00	MAE	0	0
F1-Sscore	1.00	1.00	RMSE	0	0.0
Accuracy	1.00	1.00	G-Mean	1.00	1.00
Brier score loss	0	0	MCC	1.00	1.00
TPR	1.00	1.00	CK	1.00	1.00
TNR	1.00	1.00	Log_loss	0	0

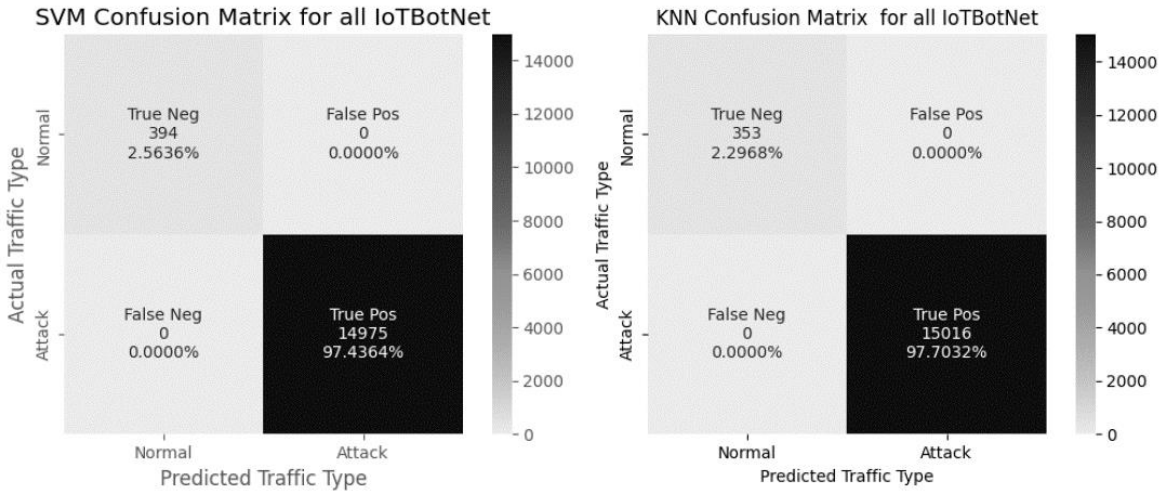

Figure 4.9 Confusion matrix for all UNSW-Bot-IoT 2018 attack (KNN and SVM)

4.5.3.3 Evaluation on Real-time

After training and testing the algorithms on two types of databases containing different and modern DDoS attacks, the proposed system is now tested in real time using this dataset as a benchmark dataset.

On the other hand, the Raspberry Pi 400, with an IP address of 192.168.0.10, sends data normally to the fog layer. On the other hand, the Raspberry Pi 400, with an IP address of 192.168.0.9, generates DDoS attacks using scapy tools and Hping3 as shown in Figure (4.10).

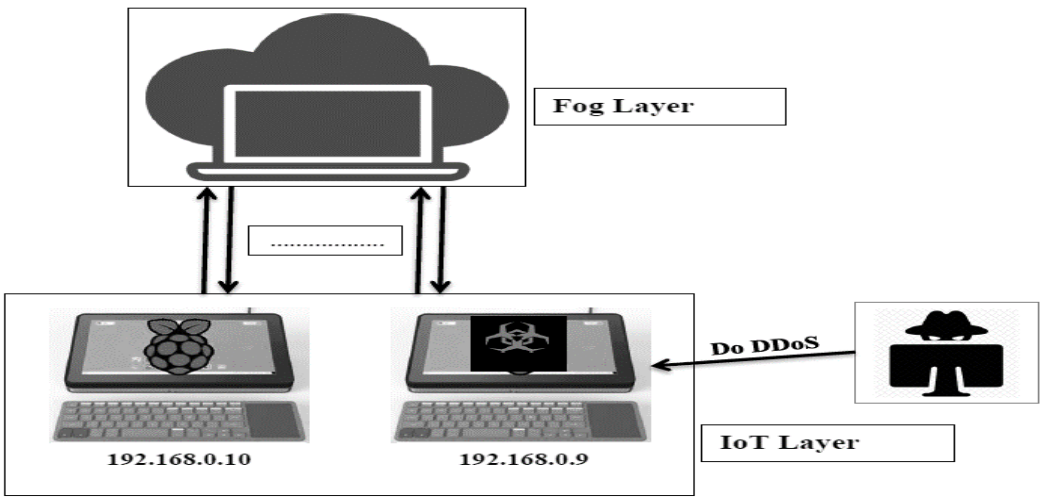

Figure 4.10 An overview of how the system works in real time.

Using the Python programming language's scapy library, it is possible to create various DDoS attacks on the TCP and UDP protocols. Using Ubuntu terminals, Hping3 also generates very potent distributed denial of service attacks.

Five types of DDoS attacks were applied to the proposed system to test its accuracy in detection and mitigation. Each attack was implemented and evaluated separately from the others. The time limit for evaluation of the proposed system is 2 hours and 20 minutes. During this time, different DDoS attacks were carried out at different times.

Before discussing the specifics and types of each attack, the network must be displayed and monitored in terms of throughput and network monitor (uploading and downloading):

1. Throughput

Normal network operation is depicted in Figure (4.11) as the transmission of a message containing a single reading from the Raspberry Pi to the fog layer in one second.

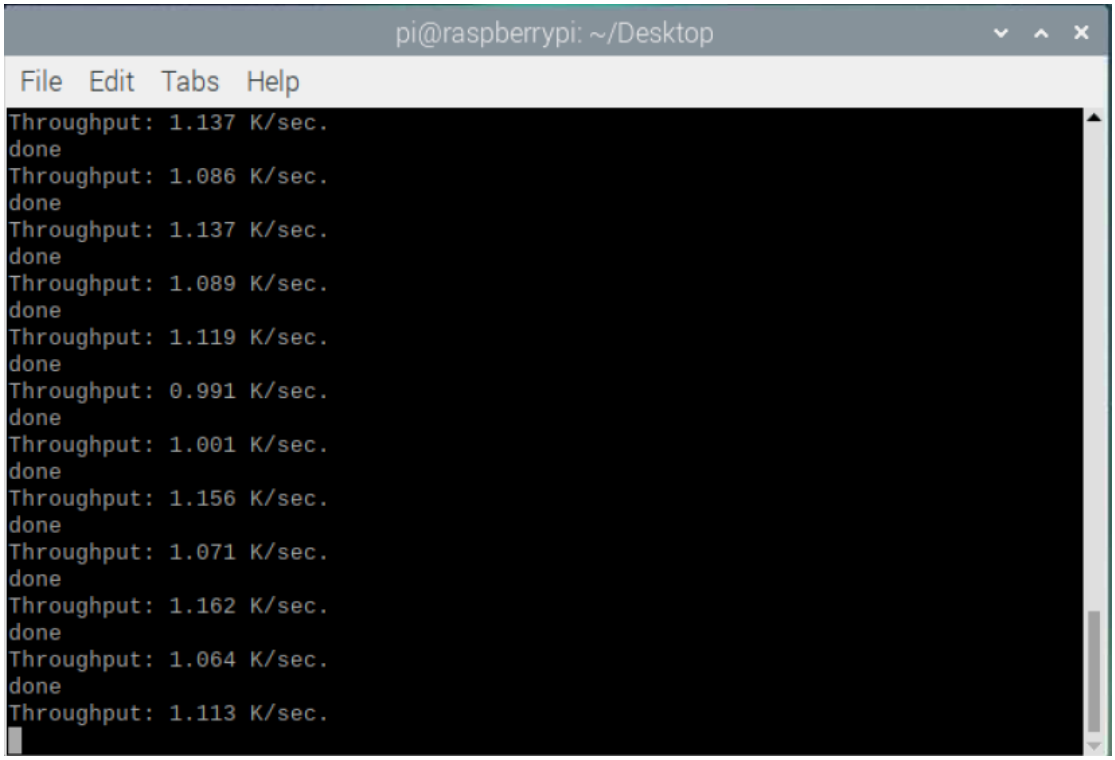

Figure 4.11 throughput in normal state.

As depicted in Figure (4.12), in the event of a distributed denial of service attack, the network throughput drops significantly, reaching a transmission rate of 0.08 KB per second.

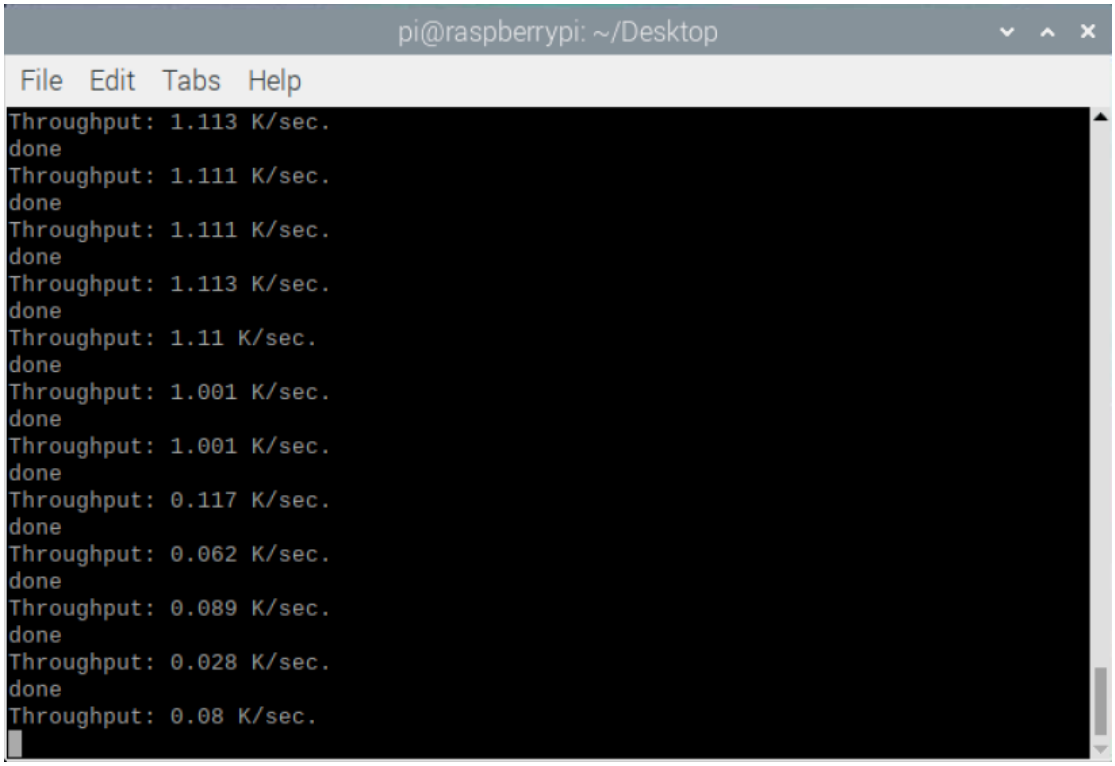

Figure 4.12 throughput in normal state.

2. Network monitor

Upload speed refers to the pace at which data is transmitted, whereas download speed refers to the rate at which data is received. In the normal state of the network, these readings are different from in the attack state. As demonstrated in Figure (4.13-b), the read download and delivery rates are quite high in the event of an attack.

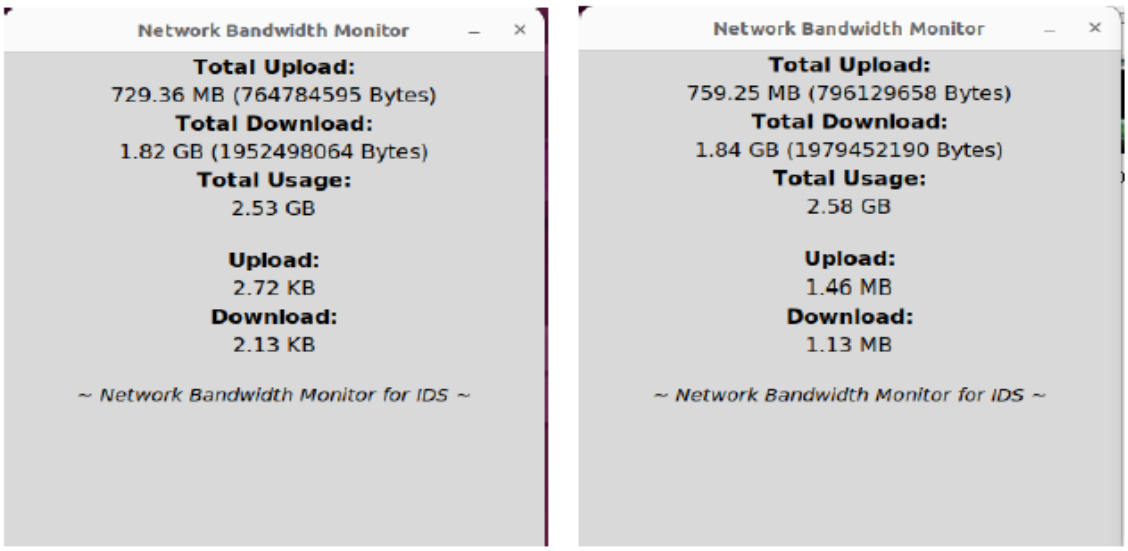

Figure 4.13 a-normal network traffic. b-attack network traffic.

Following the presentation of the current state of the network, will now provide the primary user interface of the DDoS attack detection system that have suggested.

Following the execution of the proposed system, it was subjected to testing with the scapy and Hping3 library, which involved five different kinds of assaults (HTTP, DNS, LDAP, MSSQL, and NetBIOS). As shown in Table (4.21), which shows the attack methods using Hping3, in addition to the algorithm (4.1), which shows the attack method using the scapy tools.

Table 4.21 Hping DDoS attack

Attack type	Script	Protocol
HTTP	Sodu hping3 –s –flood –rand-source –p 80 192.168.0.100	TCP
DNS	Sodu hping3 --udp –flood –rand-source –p 53 192.168.0.100	UDP

LDAP	Sodu hping3 –s –flood –rand-source –p 636 192.168.0.100	TCP
MSSQL	Sodu hping3 –s –flood –rand-source –p 1433 192.168.0.100	TCP
NetBIOS	Sodu hping3 --udp –flood –rand-source –p 138 192.168.0.100	UDP

Algorithm 4.1 Scapy tools DDoS attack

Input (target IP, target Port)

Output flood paket

While True **Do**
Tcp(
IPlayer = IP(src=random ,dst= target IP)
TCPlayer = TCP(sport=random, target Port)
pkt = IPlayer / TCPlayer
send(pkt))
UDP(
IPlayer = IP(src=random ,dst= target IP)
UDPlayer = UDP(sport=random, target Port)
pkt = IPlayer / UDPlayer
send(pkt)
)

In the proposed system, for each network reading, the reading is compared to a threshold value, If the Source IP's entropy value is less than the threshold value, the data is classed as normal; otherwise, the reading is sent to machine learning algorithms for decision-making. In order for the algorithms to agree on a threshold value, traffic is classed as suspicious if one of the algorithms agrees with the entropy value indicating that the data is suspicious, as depicted in Figure (4.14).

98	3.308271	3.7946534	0.7642045	1.8428574	1	[1]	[1]	1
99	3.7229285	4.1499963	0.712064	1.7474762	1	[1]	[1]	1
100	3.398114	3.8713508	0.753198	1.8240126	1	[1]	[1]	1
101	2.321166	2.9278374	1.4158211	2.3149257	1	[0]	[1]	1
102	3.2345295	3.55309	0.7495953	1.7307544	1	[1]	[1]	1
103	3.2915626	3.587859	0.7642045	1.7320914	1	[1]	[1]	1
104	3.4508157	4.0159206	0.8954864	2.0735738	1	[1]	[1]	1

Figure 4.14 Algorithms vote with entropy to confirm the attack.

The recent results of the proposed system for detecting the five types of real-time DDoS attacks are presented in Table (4.22).

Table 4.22 Evaluation propose system on real-time.

Evaluation metrics	**KNN**	**SVM**	**Evaluation metric**	**KNN**	**SVM**
Recall	1.00	1.00	FPR	0.0017	0.0022
			FNR	0	0
Precision	1.00	1.00	MAE	0.0009	0.00118
F1-Sscore	1.00	1.00	RMSE	0.0313	0.0343
Accuracy	0.9990	0.9988	G-Mean	0.9991	0.9988
Brier score loss	0.0009	0.0002	MCC	0.9980	0.9979
TPR	1.00	1.00	CK	0.9980	0.9976
TNR	0.9982	0.9977	Log_loss	0.0340	0.04082

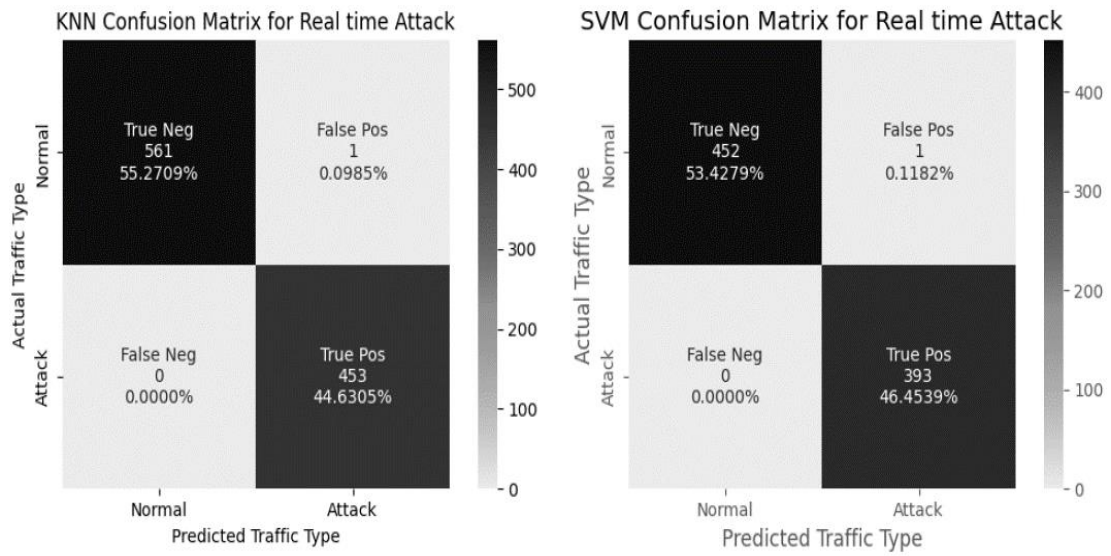

Figure 4.15 Algorithms vote with entropy to confirm the attack.

The latest data classified by the proposed system and stored in a CSV file are displayed in Table (4.22) and Figure (4.15). This data was separated into two sections: a training segment comprising 70 percent and a testing section comprising 30 percent. The results demonstrated that the suggested system achieved extremely precise classification of data in real time.

After ensuring that the proposed system is capable of detecting assaults with high precision, the mitigation phase entails putting the finishing touches in place. The suggested system for Ubuntu will utilize ufw as a straightforward firewall through which a specific port can be closed or opened using specific commands. After machine learning algorithms establish the presence of an attack, the precise port that the attack got is sent to a Python-programmed class to closes or opens the ports. It also provides the system administrator with an alert notification about the occurrence of the attack.

4.6 Comparison

The tables below presents a comparison with other works using the previously mentioned evaluation methods. In addition, it is comparable to several papers that used the UNSW-Bot-IoT 2018 and CICDDoS2019 datasets. as well as comparing it to research that works in real-time.

4.6.1 Comparison with works evaluated on CICDDoS2019

Table 4.23 Comparison propose system with other work evaluated on CICDDoS2019.

Aoth	Att.	Algo.	Acc	P	R	F1	TP	TN	FP	FN	MAE	RMSE	G-Main	MCC	CK	Log_Loss	BSL
[23]	All	Knn	0.9999	-	-	-	-	-	-	-	-	-	-	-	-	-	-
[24]	All	lstm and cnn	0.98	-	-	-	-	-	-	-	-	-	-	-	-	-	-
[25]	All	Knn	0.9128	0.98	0.95	0.94	0.94	0.94	0.04	0.19	-	-	-	-	-	-	-
[27]	Ldap	AE-MLP	-	0.91	0.99	0.95	-	-	-	-	-	-	-	-	-	-	-
	Mssql		-	0.98	0.94	0.96	-	-	-	-	-	-	-	-	-	-	-
	Netbios		-	1.00	1.00	1.00	-	-	-	-	-	-	-	-	-	-	-
	Syn		-	1.00	1.00	1.00	-	-	-	-	-	-	-	-	-	-	-
	Udp		-	0.96	0.96	0.97	-	-	-	-	-	-	-	-	-	-	-
[28]	All	Knn	0.98	0.99	0.99	0.99	-	-	-	-	-	-	-	-	-	-	-
		Svm	0.86	0.86	0.87	0.85	-	-	-	-	-	-	-	-	-	-	-
Our propose	Syn	Knn	1.00	1.00	1.00	1.00	1.00	1.00	0	0	0	0	1.00	1.00	1.00	0	0
		Svm	1.00	1.00	1.00	1.00	1.00	1.00	0	0	0	0	1.00	1.00	1.00	0	0
	udp	Knn	1.00	1.00	1.00	1.00	1.00	1.00	0	0	0	0	1.00	1.00	1.00	0	0
		Svm	0.9995	1.00	1.00	1.00	0.99	0.99	0.001	0.00 03	0.0004	0.017	0.9971	0.9916	0.9916	0.01	0.0005
	portmap	Knn	1.00	1.00	1.00	1.00	1.00	1.00	0	0	0	0	1.0	1.0	1.0	0	0
		svm	1.00	1.00	1.00	1.00	1.00	1.00	0	0	0	0	1.0	1.0	1.0	0	0
	mssql	Knn	0.9995	1.00	1.00	1.00	1.00	0.98	0.01	0	0.0004	0.021	0.9922	0.9920	0.9920	0.01	0.0005
		svm	0.9993	0.99	0,99	0,99	0.99	98	0.01	0.00 03	0.0007	0.027	0.9907	0.9844	0.9844	0.02	0.0007

	ldap	Knn	0.9990	1.00	1.00	1.00	0.99	0.98	0.01	0.006	0.0009	0.03	0.9936	0.9815	0.9815	0.03	0.0009
		svm	0.9970	1.00	1.00	1.00	0.99	0.98	0.01	0.002	0.0029	0.05	0.9899	0.9456	0.9456	0.10	0.003
	Netbios	Knn	0.9986	0.98	1.00	1.00	1.00	0.95	0.04	0	0.001	0.03	0.9771	0.9764	0.9761	0.04	0.0014
		svm	0.9986	1.00	0.99	1.00	0.99	0.98	0.01	0.001	0.001	0.04	0.9916	0.9712	0.9711	0.05	0.0014
	Udp_lag	Knn	0.9995	1.00	1.00	1.00	1.00	0.99	0.006	0	0.0004	0.020	0.9968	0.9966	0.9966	0.01	0.0005
		Svm	0.9995	1.00	1.00	1.00	0.99	1.00	0	0.00004	0.0004	0.020	0.9997	0.9970	0.9870	0.01	0.0005
	All	Knn	0.9999	1.00	1.00	1.00	0.99	1.00	0	0.00008	0.0004	0.008	0.9999	0.9986	0.9986	0.002	0.0001
		Svm	0.9999	1.00	1.00	1.00	0.99	1.00	0	0.0001	0	0.009	0.9999	0.9983	0.9983	0.003	0.0001

4.6.2 Comparison with works evaluated on UNSW-Bot-IoT 2018

Table 4.24 Comparison propose system with other work evaluated on UNSW-BOT-IoT 2018.

Aoth	Att.	Algo.	Acc	P	R	F1	TP	TN	FP	FN	MAE	RMSE	G-Main	MCC	CK	Log_Loss	BSL
[25]	http	Knn	0.99	0.99	1.00	0.99	-	-	-	-	-	-	-	-	-	0.009	-
		Svm	0.99	0.99	1.00	0.99	-	-	-	-	-	-	-	-	-	0.009	-
	Tcp	Knn	0.99	0.99	1.00	0.99	-	-	-	-	-	-	-	-	-	1.76	-
		Svm	0.99	1.00	0.99	0.99	-	-	-	-	-	-	-	-	-	5.87	-
	udp	Knn	1.00	1.00	1.00	1.00	-	-	-	-	-	-	-	-	-	4.56	-
		svm	0.99	0.99	1.00	0.99	-	-	-	-	-	-	-	-	-	8.93	-
[27]	all	Ae-mlp	-	0.95	0.95	0.95	-	-	-	-	-	-	-	-	-	-	-
	http	Knn	1.00	1.00	1.00	1.00	1.00	1.00	0	0	0	0	1.00	1.00	1.00	0	0

Our propose		svm	0.9988	0.99	0.99	1.00	0.99	1.00	0	0.001	0.001	0.03	0.9994	0.9751	0.9748	0.04	0.0012
	Tcp	Knn	1.00	1.00	1.00	1.00	1.00	1.00	0	0	0	0	1.00	1.00	1.00	0	0
		svm	0.9998	1.00	1.00	1.00	0.99	1.00	0	0.00001	0.0001	0.01	0.9999	0.9960	0.9960	0.006	0.0002
	Udp	Knn	1.00	1.00	1.00	1.00	1.00	1.00	0	0	0	0	1.00	1.00	1.00	0	0
		svm	1.00	1.00	1.00	1.00	1.00	1.00	0	0	0	0	1.00	1.00	1.00	0	0
	all	Knn	1.00	1.00	1.00	1.00	1.00	1.00	0	0	0	0	1.00	1.00	1.00	0	0
		svm	1.00	1.00	1.00	1.00	1.00	1.00	0	0	0	0	1.00	1.00	1.00	0	0

4.6.3 Comparison with Work evaluated in real time

Table 4.25 Comparison propose system with other work evaluated on Real-time.

Aoth	Att.	Algo.	Acc	P	R	F1	TP	TN	FP	FN	MAE	RMSE	G-Main	MCC	CK	Log_Loss	BSL
[18]	Tcp,udp and http	Knn	0.999	0.99	0.99	0.99	-	-	-	-	-	-	-	-	-	-	-
[19]	http, udp and tcp	Knn	0.97	-	-	-	-	-	-	-	-	-	-	-	-	-	-
		svm	0.98	-	-	-	-	-	-	-	-	-	-	-	-	-	-
[20]	Syn	cep	0.9310	-	-	-	0.9409	-	6.96	-	-	-	-	-	-	-	-
	udp		0.9924				0.9875	-	0.75	-	-	-	-	-	-	-	-
[22]	DDoS	svm	0.9739	-	-	0.97	-	-	-	-	-	-	-	-	-	-	-
[14]	tcp	NFV	0.9984	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Our propose	HTTP , DNS , LDAP , MSSQL and NetBIOS	knn	0.9990	1.00	1.00	1.00	1.00	0.9982	0.001	0	0.0009	0.0313	0.9991	0.9980	0.9980	0.034	0.0009
		svm	0.9988	1.00	1.00	1.00	1.00	0.9977	0.002	0	0.0011	0.0343	0.9988	0.9979	0.9976	0.040	0.0012

Chapter Five Conclusions and Future Works

5.1 Conclusions

Even though there are several technologies designed to recognize and mitigate DDoS attacks, they continue to score highly in global network security statistics. This is despite the fact that several options exist to address this issue. In commercial enterprises, intrusion detection has become a vital aspect of any protection system. Intrusion detection may be effectively integrated into computer network hardware.

An inquiry was conducted on the detection of entropy values utilizing a time-based detection method in order to determine the presence of DDoS attacks. The main conclusions points of this work were: -

1. The entropy concept gives a way to break up and reduce large amounts of data. This is accomplished by converting huge nominal datasets, such as CICDDoS2019 and Bot-IoT2018, into a numerical feature utilizing a window of consecutive packets, referred to as Window size in this study. This characteristic is useful for early detection.
2. Detecting a DDoS attack based on the entropy notion is a good strategy for studying the attack and constructing an efficient detection model using machine learning techniques.
3. A high level of accuracy and speed in classification may be achieved by selecting a subset of the features that were used to categorize the data rather than selecting all of the features.
4. The SVM algorithm gives faster attack detection compared to KNN that is why SVM is very efficient in real time.

5.2 Future Works

In this work, a proactive way based on entropy and machine learning is implemented. suggest for future work the following: -

1. Using deep learning algorithms to detect distributed denial of service attacks based on entropy.
2. Dealing with slow and weighty Denial of Service attacks.
3. Design a proactive approach to detect DDoS attacks in a short time.
4. As a DDoS avoidance approach for a big-scale company, create a massive hierarchical DDoS signature-based database using several sub-anomaly detection systems.

Reference

[1] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan, “Internet of things (IoT) security: Current status, challenges and prospective measures,” *2015 10th Int. Conf. Internet Technol. Secur. Trans. ICITST 2015*, pp. 336–341, 2016, doi: 10.1109/ICITST.2015.7412116.

[2] S. Li, L. Da Xu, and S. Zhao, “5G Internet of Things: A survey,” *J. Ind. Inf. Integr.*, vol. 10, pp. 1–9, 2018, doi: 10.1016/j.jii.2018.01.005.

[3] T. Alam, “A Reliable Communication Framework and Its Use in Internet of Things (IoT),” *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol. © 2018 IJSRCSEIT*, vol. 5, no. 10, pp. 450–456, 2018.

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” *MCC’12 - Proc. 1st ACM Mob. Cloud Comput. Work.*, pp. 13–15, 2012, doi: 10.1145/2342509.2342513.

[5] A. Rauf, R. A. Shaikh, and A. Shah, “Security and privacy for IoT and fog computing paradigm,” *2018 15th Learn. Technol. Conf. L T 2018*, pp. 96–101, 2018, doi: 10.1109/LT.2018.8368491.

[6] R. R. Brooks, I. Ozcelik, L. Yu, J. Oakley, and N. Tusing, “Distributed Denial of Service (DDoS): A History,” *IEEE Ann. Hist. Comput.*, vol. 6180, no. c, pp. 1–12, 2021, doi: 10.1109/MAHC.2021.3072582.

[7] M. Dimolianis, A. Pavlidis, and V. Maglaris, “SYN Flood Attack Detection and Mitigation using Machine Learning Traffic Classification and Programmable Data Plane Filtering,” *2021 24th Conf. Innov. Clouds, Internet Networks Work. ICIN 2021*, no. Icin, pp. 126–133, 2021, doi: 10.1109/ICIN51074.2021.9385540.

[8] D. H. Hoang and H. D. Nguyen, “A PCA-based method for IoT network traffic anomaly detection,” *Int. Conf. Adv. Commun. Technol. ICACT*, vol. 2018-Febru, pp. 381–386, 2018, doi: 10.23919/ICACT.2018.8323766.

[9] J. Foley, N. Moradpoor, and H. Ochenyi, “Employing a Machine Learning Approach to Detect Combined Internet of Things Attacks against Two Objective Functions Using a Novel Dataset,” *Secur. Commun. Networks*, vol. 2020, 2020, doi: 10.1155/2020/2804291.

[10] R. Maharaja, P. Iyer, and Z. Ye, “A hybrid fog-cloud approach for securing the Internet of Things,” *Cluster Comput.*, vol. 23, no. 2, pp. 451–459, 2020, doi: 10.1007/s10586-019-02935-z.

[11] U. Kumar, S. Navaneet, N. Kumar, and S. C. Pandey, “Isolation of DDoS Attack in IoT: A New Perspective,” *Wirel. Pers. Commun.*, vol. 114, no. 3, pp. 2493–2510, 2020, doi: 10.1007/s11277-020-07486-w.

[12] T. R. I. G. I. A. Nguyen, T. V Phan, and B. T. Nguyen, “SeArch : A Collaborative and Intelligent NIDS Architecture for SDN-based Cloud IoT Networks,” vol. X, pp. 1–18, 2019, doi: 10.1109/ACCESS.2019.2932438.

[13] F. Hussain, S. G. Abbas, M. Husnain, U. U. Fayyaz, F. Shahzad, and G. A. Shah, “IoT DoS and DDoS Attack Detection using ResNet,” *Proc. - 2020 23rd IEEE Int. Multi-Topic Conf. INMIC 2020*, 2020, doi: 10.1109/INMIC50486.2020.9318216.

[14] L. Zhou, H. Guo, and G. Deng, “A fog computing based approach to DDoS mitigation in IIoT systems,” *Comput. Secur.*, vol. 85, pp. 51–62, 2019, doi: 10.1016/j.cose.2019.04.017.

[15] B. A. Khalaf, S. A. Mostafa, A. Mustapha, M. A. Mohammed, and W. M. Abduallah, “Comprehensive review of artificial intelligence and statistical approaches in distributed denial of service attack and defense methods,” *IEEE Access*, vol. 7, pp. 51691–51713, 2019, doi: 10.1109/ACCESS.2019.2908998.

[16] S. Ghazanfar, F. Hussain, A. U. Rehman, U. U. Fayyaz, F. Shahzad, and G. A. Shah, “IoT-Flock: An Open-source Framework for IoT Traffic Generation,” *2020 Int. Conf. Emerg. Trends Smart Technol. ICETST 2020*, 2020, doi: 10.1109/ICETST49965.2020.9080732.

[17] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, “Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy,” *Proc. - Int. Carnahan Conf. Secur. Technol.*, vol. 2019-Octob, no. Cic, 2019, doi: 10.1109/CCST.2019.8888419.

[18] R. Doshi, N. Apthorpe, and N. Feamster, “Machine learning DDoS detection for consumer internet of things devices,” *Proc. - 2018 IEEE Symp. Secur. Priv. Work. SPW 2018*, no. Ml, pp. 29–35, 2018, doi: 10.1109/SPW.2018.00013.

[19] K. Gurulakshmi and A. Nesarani, “Analysis of IoT Bots against DDOS attack using Machine learning algorithm,” *2018 2nd Int. Conf. Trends Electron. Informatics*, no. Icoei, pp. 1052–1057, 2018, doi: 10.1109/ICOEI.2018.8553896.

[20] A. M. Da Silva Cardoso, R. F. Lopes, A. S. Teles, and F. B. V. Magalhaes, “Real-time DDoS attack detection based on Complex Event Processing for IoT,” *Proc. - ACM/IEEE Int. Conf. Internet Things Des. Implementation, IoTDI 2018*, pp.

273–274, 2018, doi: 10.1109/IoTDI.2018.00036.

[21] R. K. Shrivastava, B. Bashir, and C. Hota, *Attack detection and forensics using honeypot in IoT environment*, vol. 11319 LNCS. Springer International Publishing, 2019. doi: 10.1007/978-3-030-05366-6_33.

[22] J. V. Cardoso, H. V. Sampaio, C. A. Souza, and C. B. Westphall, "DoS attack detection and prevention in fog-based intelligent environments," *Brazilian Journal of Development*, vol. 5, no. 11. pp. 23934–23956, 2019. doi: 10.34117/bjdv5n11-089.

[23] R. A. Shaikh, S. R. Hassan, and M. A. Lawal, "A DDoS Attack Mitigation Framework for IoT Networks using Fog Computing," *Procedia Comput. Sci.*, vol. 182, pp. 13–20, 2020, [Online]. Available: https://doi.org/10.1016/j.procs.2021.02.003

[24] B. S. Bishnoi, Shubham, Sagarika Mohanty, "A Deep Learning-Based Methodology in Fog Environment for DDOS Attack Detection," no. Iccmc, pp. 201–206, 2021.

[25] V. Gaur and R. Kumar, "Analysis of Machine Learning Classifiers for Early Detection of DDoS Attacks on IoT Devices," *Arab. J. Sci. Eng.*, vol. 47, no. 2, pp. 1353–1374, 2022, doi: 10.1007/s13369-021-05947-3.

[26] A. Churcher *et al.*, "An experimental analysis of attack classification using machine learning in IoT networks," *Sensors (Switzerland)*, vol. 21, no. 2, pp. 1–32, 2021, doi: 10.3390/s21020446.

[27] T. Liu, F. Sabrina, J. Jang-Jaccard, W. Xu, and Y. Wei, "Artificial intelligence-enabled ddos detection for blockchain-based smart transport systems," *Sensors*, vol. 22, no. 1, pp. 1–22, 2022, doi: 10.3390/s22010032.

[28] R. J. Alzahrani and A. Alzahrani, "Security analysis of ddos attacks using machine learning algorithms in networks traffic," *Electron.*, vol. 10, no. 23, 2021, doi: 10.3390/electronics10232919.

[29] P. Kumar, R. Kumar, G. P. Gupta, and R. Tripathi, "A Distributed framework for detecting DDoS attacks in smart contract-based Blockchain-IoT Systems by leveraging Fog computing," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 6, pp. 1–31, 2021, doi: 10.1002/ett.4112.

[30] Y. Harbi, Z. Aliouat, S. Harous, A. Bentaleb, and A. Refoufi, "A Review of Security in Internet of Things," *Wirel. Pers. Commun.*, vol. 108, no. 1, pp. 325–344, 2019, doi: 10.1007/s11277-019-06405-y.

[31] C. Li, J. Zhang, X. Yang, and L. Youlong, “Lightweight blockchain consensus mechanism and storage optimization for resource-constrained IoT devices,” *Inf. Process. Manag.*, vol. 58, no. 4, 2021, doi: 10.1016/j.ipm.2021.102602.

[32] E. S. Priya, “Big Data: Analytics, Technologies, and Applications,” *Evol. Bus. Cyber Age*, pp. 153–171, Mar. 2020.

[33] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications,” *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015, doi: 10.1109/COMST.2015.2444095.

[34] R. Buyya and S. N. Srirama, “Fog and edge computing : principles and paradigms,” *WILEY*, p. 512, 2019.

[35] H. Ning and Z. Wang, “Future Internet of Things Architecture : Like Mankind Neural System or Social Organization Framework?,” *IEEE Commun. Lett.*, vol. 15, no. 4, pp. 461–463, 2011, doi: 10.1109/LCOMM.2011.022411.110120.

[36] M. Kalmeshwar and A. P. D. N. P. K S, “Internet Of Things: Architecture,Issues and Applications,” *Int. J. Eng. Res. Appl.*, vol. 07, no. 06, pp. 85–88, 2017, doi: 10.9790/9622-0706048588.

[37] I. Mashal, O. Alsaryrah, T. Y. Chung, C. Z. Yang, W. H. Kuo, and D. P. Agrawal, “Choices for interaction with things on Internet and underlying issues,” *Ad Hoc Networks*, vol. 28, no. January, pp. 68–90, 2015, doi: 10.1016/j.adhoc.2014.12.006.

[38] K. Bhardwaj, J. C. Miranda, and A. Gavrilovska, “Towards IoT-DDoS prevention using edge computing,” *USENIX Work. Hot Top. Edge Comput. HotEdge 2018, co-located with USENIX ATC 2018*, 2018.

[39] S. Sicari, A. Rizzardi, D. Miorandi, and A. Coen-Porisini, “REATO: REActing TO Denial of Service attacks in the Internet of Things,” *Comput. Networks*, vol. 137, pp. 37–48, 2018, doi: 10.1016/j.comnet.2018.03.020.

[40] A. Munshi, N. A. Alqarni, and N. Abdullah Almalki, “DDOS Attack on IOT Devices,” *ICCAIS 2020 - 3rd Int. Conf. Comput. Appl. Inf. Secur.*, pp. 5–9, 2020, doi: 10.1109/ICCAIS48893.2020.9096818.

[41] D. Mendez Mena, I. Papapanagiotou, and B. Yang, “Internet of things: Survey on security,” *Inf. Secur. J.*, vol. 27, no. 3, pp. 162–182, 2018, doi: 10.1080/19393555.2018.1458258.

[42] P. P. Ray, “A survey on Internet of Things architectures,” *J. King Saud Univ. -*

Comput. Inf. Sci., vol. 30, no. 3, pp. 291–319, 2018, doi: 10.1016/j.jksuci.2016.10.003.

[43] P. J. Beslin Pajila and E. Golden Julie, "Detection of DDoS Attack Using SDN in IoT: A Survey," *Lect. Notes Data Eng. Commun. Technol.*, vol. 33, pp. 438–452, 2020, doi: 10.1007/978-3-030-28364-3_44.

[44] R. Khader and D. Eleyan, "Survey of DoS/DDoS attacks in IoT," *Sustain. Eng. Innov.*, vol. 3, no. 1, pp. 23–28, 2021, doi: 10.37868/sei.v3i1.124.

[45] M. M. Salim, S. Rathore, and J. H. Park, *Distributed denial of service attacks and its defenses in IoT: a survey*, vol. 76, no. 7. Springer US, 2020. doi: 10.1007/s11227-019-02945-z.

[46] J. Y. Khan, "Introduction to IoT Systems," *Internet of Things (IoT)*, pp. 1–24, 2019, doi: 10.1201/9780429399084-1.

[47] B. Kepceoglu, A. Murzaeva, and S. Demirci, "Performing energy consuming attacks on IoT devices," *27th Telecommun. Forum, TELFOR 2019*, pp. 19–22, 2019, doi: 10.1109/TELFOR48224.2019.8971102.

[48] N. F. Syed, Z. Baig, A. Ibrahim, and C. Valli, "Denial of service attack detection through machine learning for the IoT," *J. Inf. Telecommun.*, vol. 4, no. 4, pp. 482–503, 2020, doi: 10.1080/24751839.2020.1767484.

[49] M. Dulik, "Network attack using TCP protocol for performing DoS and DDoS attacks," *2019 Commun. Inf. Technol. Conf. Proceedings, KIT 2019 - 10th Int. Sci. Conf.*, pp. 1–6, 2019, doi: 10.23919/KIT.2019.8883481.

[50] Z. Kanmai, "TCP/IP protocol security problems and defenses," *Proc. - 2020 Int. Conf. Intell. Comput. Human-Computer Interact. ICHCI 2020*, pp. 117–120, 2020, doi: 10.1109/ICHCI51889.2020.00033.

[51] Q. Chen, H. Chen, Y. Cai, Y. Zhang, and X. Huang, "Denial of Service Attack on IoT System," *Proc. - 9th Int. Conf. Inf. Technol. Med. Educ. ITME 2018*, pp. 755–758, 2018, doi: 10.1109/ITME.2018.00171.

[52] Sanjay, B. Rajendran, and P. Shetty, "DNS Amplification DNS Tunneling Attacks Simulation, Detection and Mitigation Approaches," *Proc. 5th Int. Conf. Inven. Comput. Technol. ICICT 2020*, pp. 230–236, 2020, doi: 10.1109/ICICT48043.2020.9112413.

[53] K. Ozdincer and H. A. Mantar, "SDN-based Detection and Mitigation System for DNS Amplification Attacks," *3rd Int. Symp. Multidiscip. Stud. Innov. Technol. ISMSIT 2019 - Proc.*, no. Figure 2, 2019, doi:

10.1109/ISMSIT.2019.8932809.

[54] J. R. de Almeida Neto, L. S. Souza, and A. de Ribamar Lima Ribeiro, "Comparative analysis between the k-means and fuzzy c-means algorithms to detect UDP flood DDoS attack on a SDN/NFV environment," *WEBIST 2020 - Proc. 16th Int. Conf. Web Inf. Syst. Technol.*, no. Webist, pp. 105–112, 2020, doi: 10.5220/0010176201050112.

[55] A. R. Shaaban, E. Abdelwaness, and M. Hussein, "TCP and HTTP Flood DDOS attack analysis and detection for space ground network," *2019 IEEE Int. Conf. Veh. Electron. Safety, ICVES 2019*, pp. 1–6, 2019, doi: 10.1109/ICVES.2019.8906302.

[56] J. D. Gadze, A. A. Bamfo-Asante, J. O. Agyemang, H. Nunoo-Mensah, and K. A.-B. Opare, "An Investigation into the Application of Deep Learning in the Detection and Mitigation of DDOS Attack on SDN Controllers," *Technologies*, vol. 9, no. 1, p. 14, 2021, doi: 10.3390/technologies9010014.

[57] L. L. Chuan, M. Roslee, K. Anuar, and P. W. Leong, "Investigation of the Algorithm of Attack on the Data Center in an IoT Network," pp. 13–18, Dec. 2021, doi: 10.1109/MICC53484.2021.9642105.

[58] F. G. Becker *et al.*, *Data Mining and Machine Learning in Cybersecurity*, vol. 7, no. 1. 2015. [Online]. Available: https://www.researchgate.net/publication/269107473_What_is_governance/link/548173090cf22525dcb61443/download%0Ahttp://www.econ.upf.edu/~reynal/Civil wars_12December2010.pdf%0Ahttps://think-asia.org/handle/11540/8282%0Ahttps://www.jstor.org/stable/41857625

[59] U. C. Kothari and M. Momayez, "Machine Learning: A Novel Approach to Predicting Slope Instabilities," *Int. J. Geophys.*, vol. 2018, no. Figure 1, 2018, doi: 10.1155/2018/4861254.

[60] J. B. You, C. McCallum, Y. Wang, J. Riordon, R. Nosrati, and D. Sinton, "Machine learning for sperm selection," *Nat. Rev. Urol.*, vol. 18, no. 7, pp. 387–403, 2021, doi: 10.1038/s41585-021-00465-1.

[61] M. A. Jabbar, S. Samreen, and R. Aluvalu, "The future of health care: Machine learning," *Int. J. Eng. Technol.*, vol. 7, no. 4, pp. 23–25, 2018, doi: 10.14419/ijet.v7i4.6.20226.

[62] A. R. Lubis, M. Lubis, and Al-Khowarizmi, "Optimization of distance formula in k-nearest neighbor method," *Bull. Electr. Eng. Informatics*, vol. 9, no. 1, pp.

326–338, 2020, doi: 10.11591/eei.v9i1.1464.

[63] A. Moldagulova and R. B. Sulaiman, “Using KNN algorithm for classification of textual documents,” in *ICIT 2017 - 8th International Conference on Information Technology, Proceedings*, 2017, pp. 665–671. doi: 10.1109/ICITECH.2017.8079924.

[64] S. Zhang, X. Li, M. Zong, X. Zhu, and D. Cheng, “Learning k for kNN Classification,” *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 3, 2017, doi: 10.1145/2990508.

[65] X. Wu *et al.*, *Top 10 algorithms in data mining*, vol. 14, no. 1. 2008. doi: 10.1007/s10115-007-0114-2.

[66] S. Zhang, D. Cheng, Z. Deng, M. Zong, and X. Deng, “A novel kNN algorithm with data-driven k parameter computation,” *Pattern Recognit. Lett.*, vol. 109, pp. 44–54, 2018, doi: 10.1016/j.patrec.2017.09.036.

[67] U. Lall and A. Sharma, “A nearest neighbor bootstrap for resampling hydrologic time series,” *Water Resour. Res.*, vol. 32, no. 3, pp. 679–693, 1996, doi: 10.1029/95WR02966.

[68] A. K. Ghosh, “On optimum choice of k in nearest neighbor classification,” *Comput. Stat. Data Anal.*, vol. 50, no. 11, pp. 3113–3123, 2006, doi: 10.1016/j.csda.2005.06.007.

[69] H. Liu, S. Zhang, J. Zhao, X. Zhao, and Y. Mo, “A new classification algorithm using mutual nearest neighbors,” *Proc. - 9th Int. Conf. Grid Cloud Comput. GCC 2010*, pp. 52–57, 2010, doi: 10.1109/GCC.2010.23.

[70] Y. P. Mack, “ Local Properties of k -NN Regression Estimates ,” *SIAM J. Algebr. Discret. Methods*, vol. 2, no. 3, pp. 311–323, 1981, doi: 10.1137/0602035.

[71] G. Baldini and D. Geneiatakis, “A performance evaluation on distance measures in KNN for mobile malware detection,” *2019 6th Int. Conf. Control. Decis. Inf. Technol. CoDIT 2019*, pp. 193–198, 2019, doi: 10.1109/CoDIT.2019.8820510.

[72] A. I. Kadhim, “Survey on supervised machine learning techniques for automatic text classification,” *Artif. Intell. Rev.*, vol. 52, no. 1, pp. 273–292, 2019, doi: 10.1007/s10462-018-09677-1.

[73] S. Mishra, P. K. Mallick, H. K. Tripathy, L. Jena, and G. S. Chae, “Stacked KNN with hard voting predictive approach to assist hiring process in IT organizations,” *Int. J. Electr. Eng. Educ.*, 2021, doi: 10.1177/0020720921989015.

[74] I. A. A. Amra and A. Y. A. Maghari, “Students Performance Prediction Using

KNN and Naïve Bayesian," in *International Conference on Advanced Computing and Communication Technologies, ACCT*, 2017, pp. 909–913. doi: 10.1109/ICITECH.2017.8079967.

[75] A. Merghadi *et al.*, "Machine learning methods for landslide susceptibility studies: A comparative overview of algorithm performance," *Earth-Science Rev.*, vol. 207, no. September 2019, 2020, doi: 10.1016/j.earscirev.2020.103225.

[76] B. Schölkopf, A. Smola, and F. Bach, *Learning with kernels: support vector machines, regularization, optimization, and beyond*٢٠٠٢ ..

[77] M. Sheykhmousa, M. Mahdianpari, H. Ghanbari, F. Mohammadimanesh, P. Ghamisi, and S. Homayouni, "Support Vector Machine Versus Random Forest for Remote Sensing Image Classification: A Meta-Analysis and Systematic Review," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 13, pp. 6308–6325, 2020, doi: 10.1109/JSTARS.2020.3026724.

[78] R. Khemchandani, Jayadeva, and S. Chandra, "Optimal kernel selection in twin support vector machines," *Optim. Lett.*, vol. 3, no. 1, pp. 77–88, 2009, doi: 10.1007/s11590-008-0092-7.

[79] G. Mountrakis, J. Im, and C. Ogole, "Support vector machines in remote sensing: A review," *ISPRS J. Photogramm. Remote Sens.*, vol. 66, no. 3, pp. 247–259, 2011, doi: 10.1016/j.isprsjprs.2010.11.001.

[80] Z. P. Fan, G. M. Li, and Y. Liu, "Processes and methods of information fusion for ranking products based on online reviews: An overview," *Inf. Fusion*, vol. 60, no. December 2019, pp. 87–97, 2020, doi: 10.1016/j.inffus.2020.02.007.

[81] B. Gomathi, R. Sujatha, and T. Padma, "Fusion of Classification with Hybrid Optimization Technique to Predict Diabetes," *Int. J. Eng. Adv. Technol.*, vol. 9, no. 3, pp. 927–929, 2020, doi: 10.35940/ijeat.c5418.029320.

[82] T. Mecheva and N. Kakanakov, "Cybersecurity in intelligent transportation systems," *Computers*, vol. 9, no. 4, pp. 1–12, 2020, doi: 10.3390/computers9040083.

[83] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Futur. Gener. Comput. Syst.*, vol. 100, pp. 779–796, 2019, doi: 10.1016/j.future.2019.05.041.

[84] G.F.S., "The Bell system technical journal," *J. Franklin Inst.*, vol. 196, no. 4, pp. 519–520, 1923, doi: 10.1016/s0016-0032(23)90506-5.

[85] I. K. Matjaz Kukar, *Machine learning and data mining: introduction to principles and algorithms*, vol. 45, no. 07. 2008. doi: 10.5860/choice.45-3834.

[86] R. Patgiri, U. Varshney, T. Akutota, and R. Kunde, "An Investigation on Intrusion Detection System Using Machine Learning," *Proc. 2018 IEEE Symp. Ser. Comput. Intell. SSCI 2018*, pp. 1684–1691, 2019, doi: 10.1109/SSCI.2018.8628676.

[87] B. Hajimirzaei and N. J. Navimipour, "Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm," *ICT Express*, vol. 5, no. 1, pp. 56–59, 2019, doi: 10.1016/j.icte.2018.01.014.

[88] E. Fayyoumi, S. Idwan, and H. Aboshindi, "Machine learning and statistical modelling for prediction of Novel COVID-19 patients case study: Jordan," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 5, pp. 122–126, 2020, doi: 10.14569/IJACSA.2020.0110518.

[89] D. Chicco, V. Starovoitov, and G. Jurman, "The Benefits of the Matthews Correlation Coefficient (MCC) over the Diagnostic Odds Ratio (DOR) in Binary Classification Assessment," *IEEE Access*, vol. 9, no. Mcc, pp. 47112–47124, 2021, doi: 10.1109/ACCESS.2021.3068614.

[90] D. Chicco, M. J. Warrens, and G. Jurman, "The Matthews Correlation Coefficient (MCC) is More Informative Than Cohen's Kappa and Brier Score in Binary Classification Assessment," *IEEE Access*, vol. 9, no. Mcc, pp. 78368–78381, 2021, doi: 10.1109/ACCESS.2021.3084050.

[91] S. Toften, S. Pallesen, M. Hrozanova, F. Moen, and J. Grønli, "Validation of sleep stage classification using non-contact radar technology and machine learning (Somnofy®)," *Sleep Med.*, vol. 75, pp. 54–61, 2020, doi: 10.1016/j.sleep.2020.02.022.

[92] P. K. Singh, "Bipolar fuzzy concept learning using next neighbor and Euclidean distance," *Soft Comput.*, vol. 23, no. 12, pp. 4503–4520, 2019, doi: 10.1007/s00500-018-3114-0.

الملخص

ساعد إدخال تقنية جديدة على النمو الهائل لإنترنت الأشياء، مما سمح بتوصيل المزيد من الأجهزة في شبكة إنترنت الأشياء من خلال توفر اتصالات أسرع وتقليل زمن الوصول.

نظرًا لأن شبكات إنترنت الأشياء (IoT) أصبحت أكثر انتشارًا ومستخدمة على نطاق واسع، فقد أصبح الأمان أحد المتطلبات الأساسية، كما أن هجوم رفض الخدمة الموزع (DDoS) يشكل تهديدًا أمنيًا كبيرًا بسبب الموارد المحدودة (وحدة المعالجة المركزية، الذاكرة، مفتوح المصدر، اتصال مستمر) يمكن استخدامه لزيادة هجمات DDOS عن قصد أو عن غير قصد. اعتداء DDoS هو هجوم قائم على الازدحام يمنع المستخدمين الشرعيين من الوصول إلى موارد الضحية عن طريق إغراقهم بحزم لا قيمة لها. يعد اكتشاف هجوم DDoS أمرًا صعبًا نظرًا لعدم وجود إرشادات موحدة لتحديد التدفق الشرعي للشبكة بشكل موثوق.

في هذا العمل، نقترح إطار عمل قائم على الانتروبيا مع خوارزميات التعلم الآلي لتحسين نتيجة الاكتشاف. يتكون النظام المقترح من ثلاثة أجزاء: الجزء الأول هو إنشاء نافذة زمنية منزلقة لحساب الانتروبيا؛ والثاني هو استخدام قيمة عتبة الانتروبيا لجعل الاكتشاف المبكر ممكنًا أثناء DDoS ولكن ليس بعد الانهيار؛ الجزء الثالث هو استخدام خوارزميات التعلم الآلي (KNN و SVM) لتحسين دقة النتائج في الكشف المبكر.

تم إجراء عدة تجارب لتنفيذ النظام المقترح. أظهرت النتائج التي تم الحصول عليها أنه قادر على اكتشاف هجمات DDoS بكفاءة عالية. كانت الدقة الإجمالية لـ CICDDoS2019 هي ٩٩,٩٩٪، وكان المعدل الإيجابي الخاطئ ٠، ومعدل فقدان السجل ٠,٠٠٣. في حين أن الدقة الإجمالية في مجموعة بيانات UNSW-Bot-IoT2018 كانت ١٠٠٪، كان المعدل الإيجابي الخاطئ ٠ وكان معدل فقدان السجل ٠. تم استخراج معلومات نمط هجوم رفض الخدمة الموزع من مجموعتي بيانات CICDDoS2019 وUNSW-Bot-IoT2018، وتم تدريب النظام المقترح عليها من أجل التنفيذ في الوقت الفعلي. كانت الدقة الكلية في الوقت الفعلي ٩٩,٨٨٪، والمعدل الإيجابي الخاطئ ٠,٠١٪، ومعدل الخسارة اللوغاريتمية ٠,٠٤٪.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل
كلية تكنولوجيا المعلومات
قسم شبكات المعلومات

كشف وتخفيف (او منع) هجمات منع الخدمة الموزع في انترنيت الاشياء باستخدام الانتروبيا وتعلم الاله في بيئة الحوسبة الضبابية

رسالة مقدمة

إلى مجلس كلية تكنولوجيا المعلومات في جامعة بابل والتي هي جزء من متطلبات

الحصول على درجة الماجستير في تكنولوجيا المعلومات / شبكات المعلومات

من قبل الطالب

كرار فالح حسن

بأشراف

أ.م.د مهدي عبادي مانع

هـ ١٤٤٤ م ٢٠٢٢