Detect and Prevent Distributed Denial of Service attacks on the Internet of Things

Dr.JULIE
M.Sc. (IT), M.Phil. (IT)
Assistant Professor
K.C College, Churchgate
vkjewani@gmail.com

Dr.HENRY
M.Sc., PGDCS, M.S. (Soft.Syst.), M.Phil. (CS), Ph.D.
Head & Associate Professor in Computer Science& Application
G S Science, Arts & Commerce College, Khamgaon, Maharashtra
Sant.Gadge Baba Amravati University, *Maharashtra*
peajmire@gmail.com, peajmire@rediffmail.com

Dr.SAI
M.Sc. (CS), M.Phil. (CS)
Assistant Professor
K.CCollege, Churchgate
geetabrijwani@gmail.com

Abstract The IoT (Internet of Things) system is an emerging area these days. It stores data in the data storage and works by exchanging network information about things. Therefore, it is very essential to focus on security of the information between the network transmissions. The most common reason for information security issues in recent years has been a DDoS attack. In this paper author proposes an Autonomous Defense System that combines edge computing with a 2D Convolutional Neural Network (CNN) that can recognize whether the IoT data server is affected by a DDoS attack and what the attack mode is. The accuracy of trained 2D CNN for packet traffic is 99.5%, and for packet features training is 99.8%, according to a field experiment. In the proposed system, the data server can easily differentiate between the attacks and normal transmissions to reduce the impact of a DDoS attack on IoT data storage when it's under attack.

Keywords: DDoS, IoT, Convolutional Neural Network; Edge Computing.

I.INTRODUCTION

Wireless Sensor Networks (WSNs) have become increasingly popular in recent years, leading to the development of the Internet of Things [1]. The purpose of IoT is to access each other's data remotely through the M2M (machine-to-machine) connection without human intervention. The IoT architecture can be broken down into three layers: the sensing layer, the network layer, and the application layer. In order to meet this heterogeneous network architecture, most IoT systems include a network layer gateway that handles heterogeneous network processing and uploads the processed data to the cloud. However, most network gateways only process heterogeneous network information, and configure a minimum amount of storage capacity. This limited storage capacity prevents users from installing anti-virus software onto IoT devices, leading to many vulnerabilities.

The rate of data interact between devices is increasing due to the network technology's quick development. Attackers can use loopholes in the system software or firmware to not only steal the information acquired through the device for selling personal information, engaging in phishing schemes, propagating spam, etc., but they can even take control of the device and execute distributed denial of service (DDoS) operations against other targets [2]. Common users may not be able to identify the device under attack or control until system or networking resources are impaired, network services are temporarily impacted, or network resources are affected for the first time. Therefore, by analyzing network packets to stop blocking attacks, we want to improve the integrity of the next IoT devices.

Numerous strategies to minimize the effect of DDoS on IoT have already been proposed in a lot of papers [3-6]. In addition, since artificial intelligence has advanced, new algorithms have been used in a variety of scenarios. Artificial intelligence's Deep Learning framework is derived from the Neural Network (NN) framework, which Hinton et al. refined in 2006 [7]. It has been used in a wide range of applications, including speech recognition, image recognition, natural language processing, etc., and it can analyze sequential data through [8–12], it can be shown that deep learning can be used to boost the efficiency of network information security detection systems and intrusion detection systems. However, the identification consequences on the edge computer are evaluated under different conditions, where Feature denotes the feature model's identification outcome and Flow denotes the traffic model's identification outcome. As a result, user carried out our investigation utilizing the relevant neural network methodology. Under normal transmission, SYN flood attack, UDP flood attack, ICMP flood attack, and MIX flood assault, respectively, the proposed architecture of edge computing with a trained CNN model will make appropriate identifications. With this suggestion in mind, we require to make the best use of information about changes in related characteristics that occur during the transmission of packets in the system, which may significantly improve recognition accuracy and mitigate for the drawbacks of utilizing a single model for judgment. The following paragraph will address some recent relevant research on machine learning (ML)-based techniques to reduce DDoS assaults in IoT systems. A comparison of the effectiveness of the Long-Short Term

Memory (LSTM) and the Random Neural Network (RNN) in detecting SYN flood DDoS attacks was made in [13]. It performs better than LSTM in RNN, but its accuracy was only about 81%, therefore it wasn't deemed good enough to rely on. A novel deep neural network to distinguish between normal and abnormal network flows was introduced in [14]. Using the most recent Canadian dataset (CIC IDS 2017), the authors implemented a feedforward back-propagation design with seven hidden layers and tested the technique for DDoS detection. The test yielded a value of 0.99 scores, indicating that the experimental findings were precise and recallable. Edge2Guard (E2G), a resource-friendly machine learning algorithm was first mentioned in [15]. The N-BaIoT dataset of regular and attack network traffic records captured through the use of Mirai and Bashlitte Botnets served as the basis for its training and testing. The approach relies on building an E2G model for each MCU-based IoT device individually in the system to have resource-friendly detection. The drawback of this technique is that, after being trained with data from the developed type of malware action, the model needs to be updated often, increasing the challenge of the deployment process. An effective method to identify two types of using two crucial features, Volumetric and Asymmetric characteristics, which are crucial for detecting two types of flooding-based DDoS assaults, are used in an efficient strategy that was given in [16]. The suggested SDN-based DDoS attack detection approach can minimize user activity disruption and cut down on training and testing time.

Additionally, it suggested using the Advanced Support Vector Machine (ASVM) technique to improve the existing Support Vector Machine (SVM) algorithm and successfully detect DDoS flooding attacks. Using SVM and CNN ML algorithms, a new detection system for classification was proposed in [17]. It converts binary files into grayscale illustrations that may be seen. The CNN and SVM then analyze these images to determine whether a file has maliciously injected code. In the case of binary classification, this method's accuracy can reach up to 94%, however in the case of multi-classification, it can only reach 81%. In order to figure out the frequency of DDoS attacks, a new ML technique based on clustering and graph structure features has been described in [18]. In graph theory, the technique generates edge and vertex configurations and extracts as input variables, eight features of traffic data as input variables the features of DDoS and regular communication are then extracted using the principal component analysis (PCA) model. Finally, DDoS is discovered using the fuzzy C-means (FCM) clustering technique. Using the 2000 traffic data from CICIDS-2017 as an example, the method's accessibility is confirmed. Recall, false positive, true positive, true negative, and false negative values were 100.00%, 1.05%, 68.95%, 0.00%, and 30.00%, respectively, showing that it outperforms other approaches in terms of detection reliability and has a favorable impact on DDoS attacks.

In the past, DDoS attacks have been avoided by locating the attack's origin and blocking it through tools like firewalls intrusion detection systems (IDS). However, the Dyn.com domain name systems (DNS) services provider, Domain Name Services, was the victim of a DDoS attack as a result of the significant rise in DDoS attacks such as the zombie infection caused by the Mirai computer virus in 2016 [19].This type of attack might be a Botnet created by numerous IoT devices generating a lot of DNS requests over a lot of different IP addresses, which would disrupt service [20]. Additionally, some attacks use masked IP addresses, making it impossible for these tactics to successfully fight against DDoS attacks. Due to a single error in judgment, it is simple to pass attack traffic or stop routine traffic. Therefore, it is essential that one figure out how to lower the error rate in DDoS attacks.

In summary, the main objective of this research is to minimize the likelihood that attack traffic packets and routine traffic packets will be misunderstood during DDoS attacks. Convolutional neural networks, also known as CNNs, are used to examine the differences between normal transmission and DDoS attacks to determine whether the current system is normal based on the collected packets.

The remaining portion of this work is organized as follows: In Section 2, a DDoS assault scenario on an IoT network is briefly described. The experimental hardware and architecture, model training dataset, model training approaches, Internet of Things architecture, DDoS attack architecture, and system detection architecture are all introduced in Section 3. The application of the suggested method and the results of the experiments come next in Section 4. The paper is finally summarized in Section 5.

II. SCENARIOS OF DDOS ATTACK

DDoS is a variant of the one-to-one transmission approach known as a Denial of Service (DoS) attack. To overwhelm the victim's network bandwidth and system resources, stop or interrupt system services, and prevent other typical users from accessing the resources they need, a huge number of imitations or meaningless packets are sent to the target machine. The DDoS attack was established because DoS attacks have become more challenging as technology for computers and network communication have advanced. It is a botnet made up of two or more compromised computers spread out around the entire world that executes DoS attacks on the same target with the objective of interrupting or discontinuing the server network service [21], as depicted in Figure 1.

2.1 DDoS Attacks and Network Bandwidth Consumption

Botnets transmit big traffic packets that use up network bandwidth and frequently block the machine being attacked.

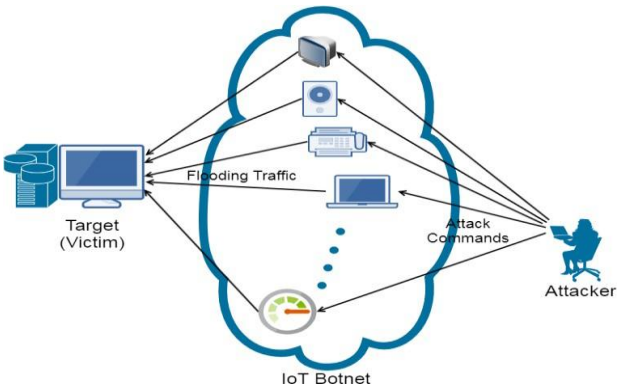


Figure 1: DDoS Attack.

2.1.1. User Datagram Protocol (UDP) Flood Attack

UDP is a transport protocol without connections. Because authentication is not required when sending packets using the UDP protocol, a lot of packets can be delivered to the victim's computer at once, overloading the bandwidth and restricting access to standard services. In Figure 2, this attack strategy is displayed.

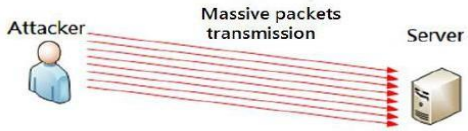


Figure 2: UDP Flood Attack

2.1.2. Internet Control Message Protocol (ICMP) Flood Attack

The ICMP echo request header packet is transmitted by the client to the master whenever the ping command is usually employed, and the ICMP echo reply to header packet is sent by the master to the client when the ping command is normally used, as demonstrated in Figure 3. However, according to in Figure 4, the ICMP flood will rapidly send several Ping commands to the targeted site, consuming up resources on the host server and degrading service.

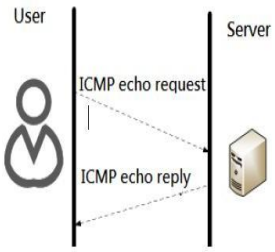


Figure 3: Normal ICMP Diagram

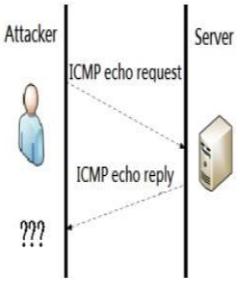


Figure 4: ICMP Flood Attack

2.1.3. Teardrop Attack

Before transmission, each packet is separated into smaller fragments and relocated, and the processing information are saved for later packet reassembly. Using this technique, the teardrop attack will intentionally shift information, making it challenging for the packet to be correctly reassembled and resulting in errors.

2.2. System Resource Consumption DDoS Attacks

System transmission flaws or fake IPs are the primary causes of system resource consumption, which eventually results in the suspension or end of service.

2.2.1. Synchronize (SYN) Flood Attack

Packet ACK The server will keep sending SYN + ACK packets until user reply, at the moment it times out. In turn, this takes up memory and bandwidth on the server itself. The three-way handshake procedure during normal TCP transmission is depicted in the following picture, as seen in picture 5. The three-way handshake mechanism of TCP transmission under SYN flood attacks is depicted in Figure 6. Attacks that are called Local Area Network Denial (LAND) the primary distinction between SYN flood and it is that the attacked host's IP is used as the fake IP. As a result, the host enters an infinite loop and keeps using the resources of the targeted host through the transmission of SYN + ACK packets back to itself.

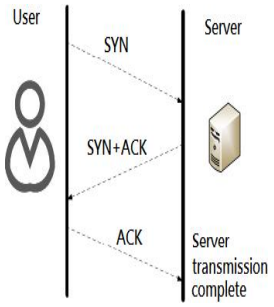


Figure 5: TCP three way Handshake process

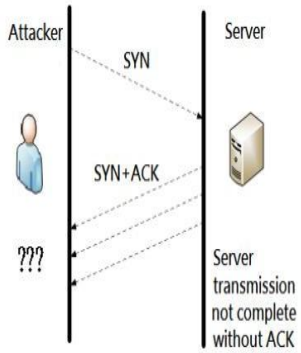


Figure 6: SYN Flood Attack

2.2.2. Local Area Network Denial (LAND) Attack

The primary distinction between SYN flood and it is that the attacked host's IP is used as the fake IP. As a result, the host creates an infinite loop and keeps using the resources of the targeted host through the transmission of SYN + ACK packets back to itself.

2.2.3. DNS Flood Attack

A network attack on the DNS is called a DNS flood. By leveraging botnets to deliver randomly generated DNS requests to the DNS server, the server is unable to locate the necessary subdomain names, which disrupts DNS functionality.

3. System Model

The experimental hardware and architecture, ML model training dataset, ML model training approaches, IoT architecture, DDoS attack architecture, and system detection architecture used in this research are all introduced in this part.

3.1. Experimental Hardware and Environment Architecture

In this study, the wireless network card (2.4 GHz 802.11n/5 GHz 802.11ac) communicates data to the PC at the edge of computing while a Raspberry Pi 3B (802.11n) serves as the main server for IoT transmission. A Raspberry Pi 3B and a Raspberry Pi 3B + (802.11b/g/n/ac) are used for the DDoS attack side. Transmission takes place between the two. System attacks on the data collection end will be carried out using 2.4 GHz Wi-Fi, as well as attacks on the network using Wi-Fi itself. The tiny D1 and humidity and temperature sensors are used as an Internet of Things (IoT) sensor node to collect environment sensor data and send it over Wi-Fi at 2.4 GHz. The local end of edge computing, which checks for anomalous transmissions in the data gathering server, is a personal computer. In this study, utilizing edge operation is done mainly to lessen the Raspberry Pi system's judgment mistake and the transmission latency imposed on by DDoS attacks. As a result, the useful transmission technique for edge operation uses 5 GHz Wi-Fi in different network domains from the data sensing node. As shown in the example below, this paper will employ AP1 and AP2 as two Wireless Access Points (AP). AP1 uses a frequency of 2.4 GHz, while AP2 uses a frequency of 5 GHz for Wi-Fi. The Table 1 Wi-Fi specifications list.

Table 1. IEEE 802.11 specifications.

Standard	Frequency (GHz)	Bandwidth (MHz)	TX Rate Mbit/s	MIMO
IEEE 802.11	2.4	20	2	NA
IEEE 802.11a	5	20	50	NA
IEEE 802.11b	2.4	20	10	NA
IEEE 802.11g	2.4	20	50	NA
IEEE 802.11n	2.4/5	40	130	3
IEEE 802.11ac	2.4	160	850 (Single stream)	8
IEEE 802.11ax	2.4/5/6	160	1100 (single stream)	8

3.2. Training Dataset and Model Training

In this study, the Raspberry Pi is equipped with an additional Wi-Fi wireless network card that collects data from the sensing nodes so that it can have two IP addresses at once: IP1, the Raspberry Pi's IP for wireless transmission, and IP2, the IP provided by the wireless card. The use of IP1 and IP2 shall follow the clarification provided below. With a WiFi frequency of 2.4 GHz, IP1 is utilized to receive data from the sensing node, whereas IP2 is solely used to connect to the edge computing machine. The experimental dataset for this study is comprised of the number of packets sent per second and the characteristics of each packet flow throughout the DDoS attack on the server.

3.2.1. Packet Traffic Capture Dataset

Based on the findings of the research in [22], this dataset was created. Attacks will have an impact on the system hardware's CPU and memory utilization rates. By include TCP, UDP, ICMP, and other transmission packets, every bit of packet traffic per second will be counted, and the CPU and memory consumption rates under monitoring will be identified. In addition, the following are noted: regular transmission, SYN flood attack, UDP flood attack, ICMP flood attack, and MIX flood attack. The system is restarted to make sure the data is not impacted by the previous attack testing during each test, which continues for two hours. Diagram of the packet traffic features delivery and capture is shown in figure 7.

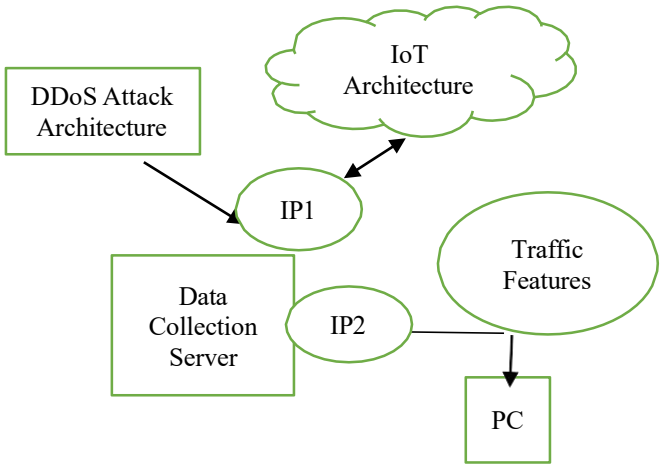


Figure 7: Schematic diagram of packet traffic features capturing and packet delivering.

The CPU and memory utilization rates, the quantity of TCP, UDP, ICMP, and other packets, as well as their counts, are among the packet traffic features that can be obtained directly from packet information. Table 2 displays the packet traffic dataset for typical transmission.

Table 2. The packet traffic dataset

Memory	CPU	No. TCP	No. UDP	No. ICMP	No. Other
33.6451895	6.1	0	1	0	0
33.6638273	12.2	0	2	0	0
33.6333890	0	0	1	0	0
33.6329414	6.5	10	2	0	0
33.6387604	0	10	1	0	0
33.6324937	0	10	2	0	0
33.6602463	5.5	10	1	0	0
33.6602463	1	10	2	0	0
33.66.2463	6.5	10	1	0	0
33.6338366	10.8	10	2	0	0

3.2.2. Packet Features Capture Dataset

Based on the outcomes of the study published in [22], this dataset was created. According to Figure 8, it employs TShark to capture each packet's information as it passes through IP1 and subsequently sends the information to the computing edge devices over IP2. The machine will separate the method of packet transmission, the amount of time between subsequent packet transmissions, the sequence number, and the size of the captured transmission packet from the captured data. Figure 9 depicts the edge computing computer's pre-processing procedure for collecting packet features. Directly extracted from the packet information are the sequence number, size, and time elapsed between two packets (interval). Under certain circumstances, information will be filled with 0 if the field in the packet is empty. For TCP, UDP, and ICMP, the transmission mode (TX mode) will be coded as 0, 1, and 2, correspondingly. Figure 10 displays the preprocessing outcome from a packet of information. Table 3 displays the packet features dataset during normal transmission.

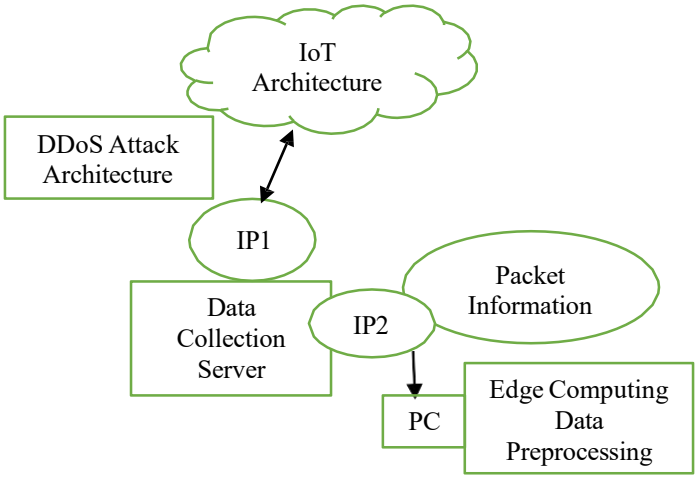


Figure 8: Schematic diagram of packet features capturing and packet delivering for edge computing.

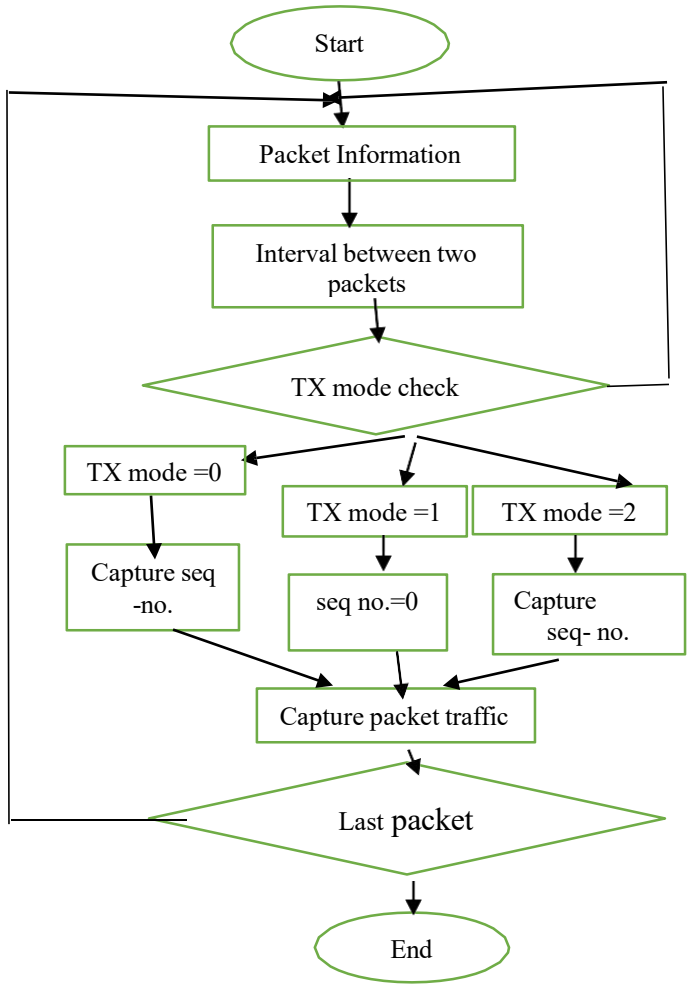


Figure 9: The Procedure of data preprocessing for packet features capturing.

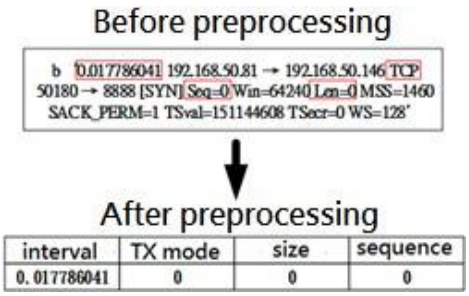


Figure 10: The result of data pre-processing For packet feature capture.

Table 3. The packet feature Dataset

Interval	TX Mode	Size	Sequence
1.011111562	1	4	0
0.000112396	2	4	0
0.500199167	0	16	1
0.001090100	0	0	1
0.000265677	0	6	1
0.004458860	0	0	7
0.009158385	0	0	17
0.000061667	0	0	8
0.000308073	0	0	0
0.000065312	0	0	0

3.3. IoT Architecture

The sensor node used in the present research is a temperature and humidity sensor (DHT11). As seen in Figure 11, the data is transmitted using the D1 mini, IP1, and AP1 for WiFi connection at a frequency of 2.4 and is then saved on the server. Figure 12 illustrates the transmission procedure.

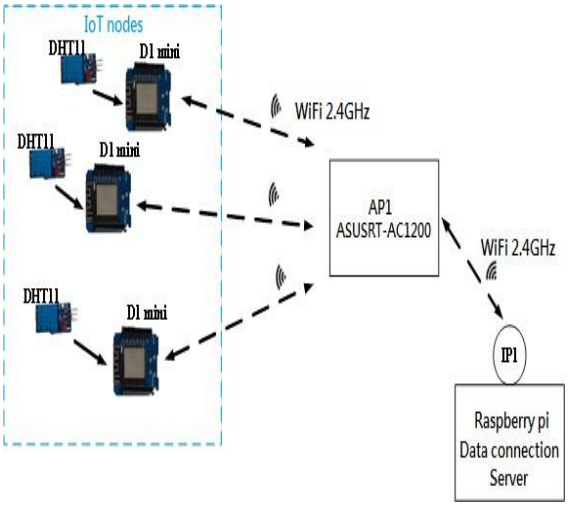


Figure 11: The architecture of IoT in real experiment system.

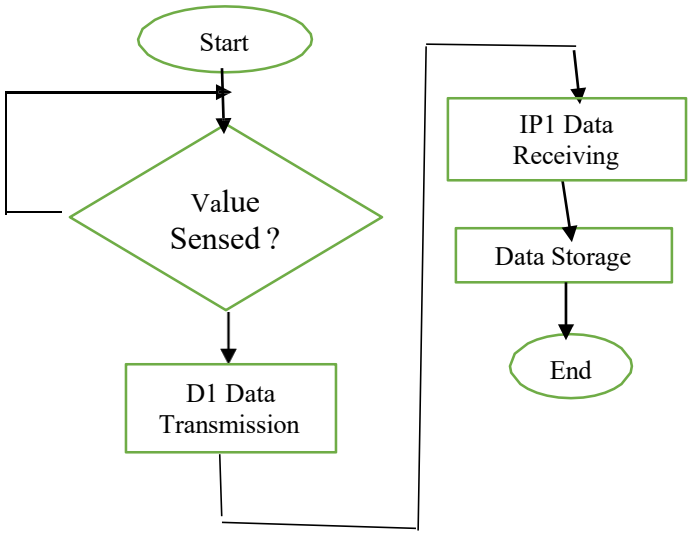


Figure 12: Diagram of the transmission process.

3.4. DDoS Attack Architecture

This paper sets up a DDoS attack situation for collecting information on the IoT server indicate during a DDoS assault. TFN2K, a DDoS simulator, is the DDoS tool employed in this study to imitate the server attack. The tool can launch SYN flood attacks, UDP flood attacks, ICMP flood attacks, and combined attacks (MIX flood) on the server. Figures 13 and 14 depict its architecture. Through control commands, this utility can manage more machines to create a botnet for DDoS attacks. As a result, this article collects data using the tool's four available methods of attack as well as standard transmission. There are a total of 4.5 million messages for packet feature data capture and 25,000 messages for collecting packet traffic data.

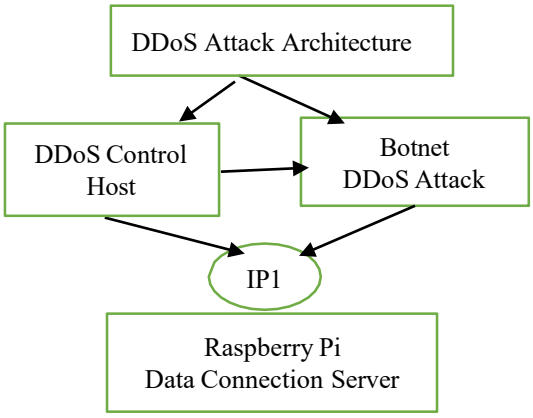


Figure 13: DDos attack for practical evaluation system

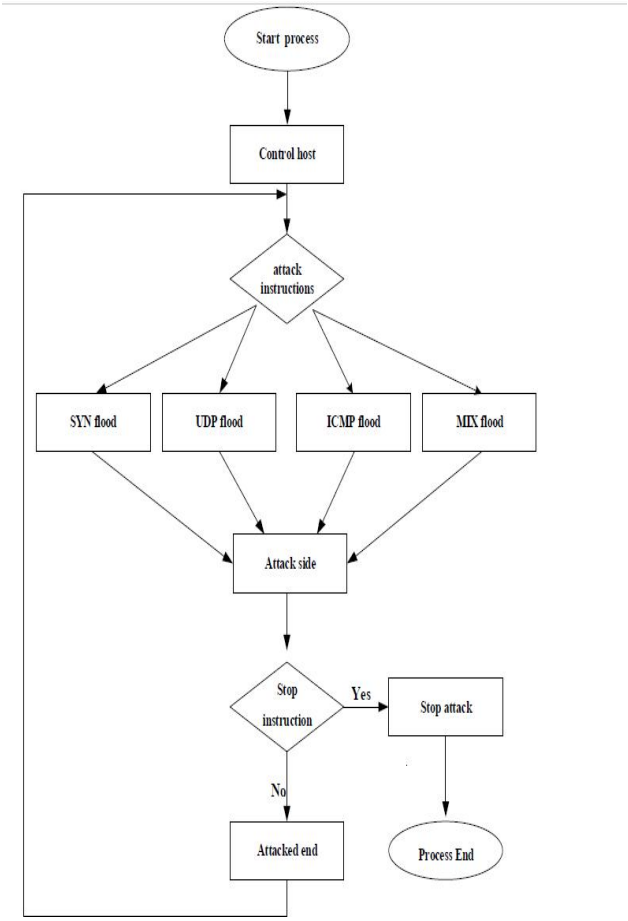


Figure 14: The procedure of DDoS attack

3.5. System Detection Architecture

To determine whether the system is abnormal, this paper system detection method involves intercepting packet traffic going through IP1, analysing the packet properties, and then transmitting the results to an edge computing device via IP2. IP1 is highly vulnerable to attack because it is linked to AP1 for the transmission of the IoT system. To mimic an attack on the server, this study will attack IP1. A wireless network card is added to the data server in this paper, which is connected to AP2 to communicate with the edge, due to flaws in the Raspberry Pi hardware system that could cause misjudgement. Computer. Consequently, IP2 refers to the IP of the new network card, and IP2 is only used for interacting with cutting-edge computers. In Figure 15, the detecting procedure is displayed.

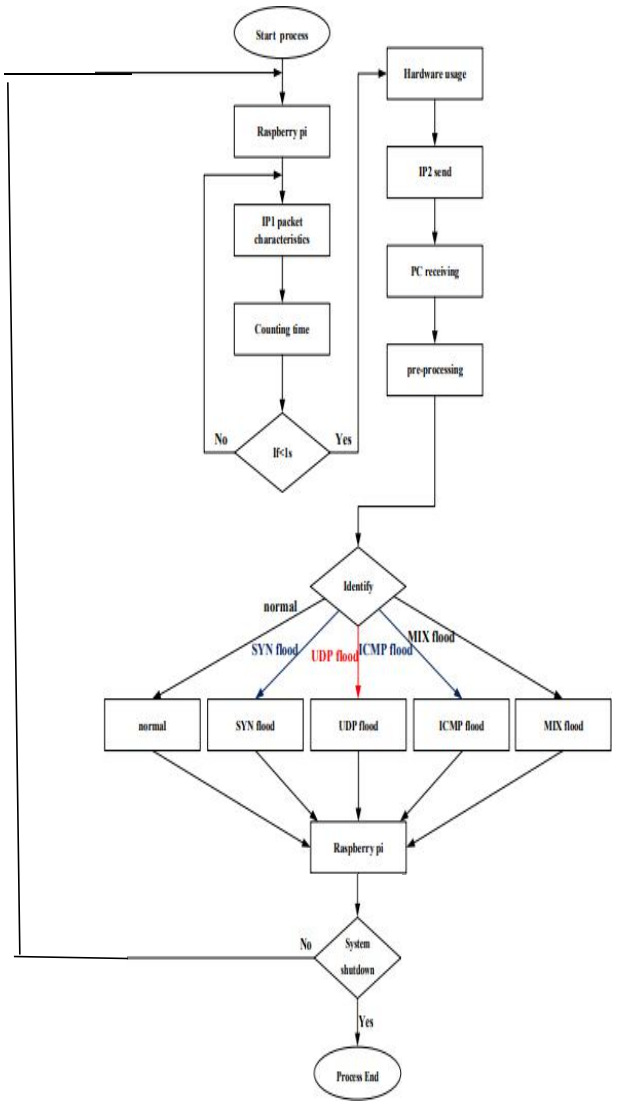
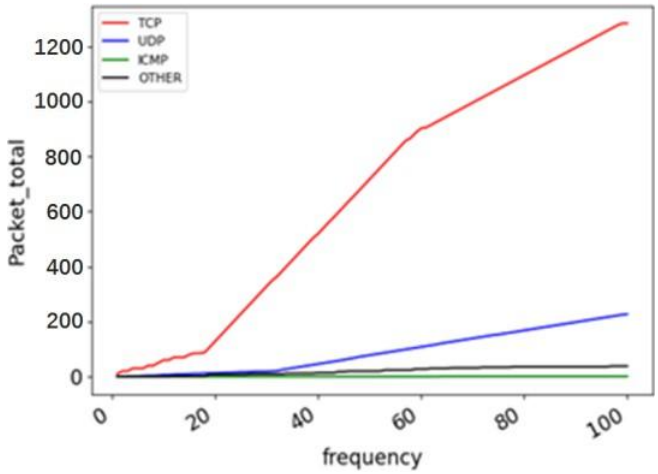


Figure 15: Procedure from packet receiving to edge computing.

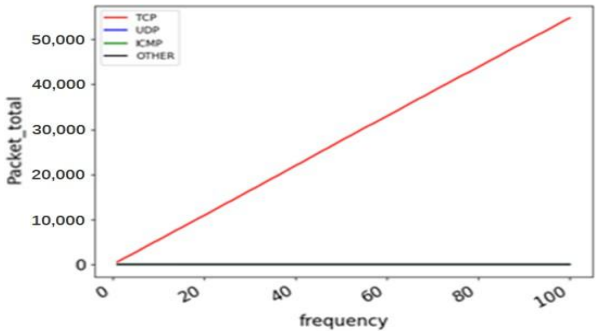
4. Results and Discussion

For five different scenarios—normal transmission, SYN flood attack, UDP flood attack, ICMP flood attack, and MIX flood attack—TFN2K is utilized in this study to gather statistics on packet traffic, CPU and memory utilization, average network speed, and other characteristics. The gathered information is pre-processed to provide two datasets, one for packet traffic and the other for packet features, which can be used to train artificial intelligence models. In this study, the packet traffic is observed for five minutes, and the pertinent characteristics of the traffic are recorded. The total number of TCP, UDP, ICMP, and other packets are shown in Figure 16a-e during normal transmission as well as during SYN flood attack, UDP flood attack, ICMP flood attack, and MIX flood

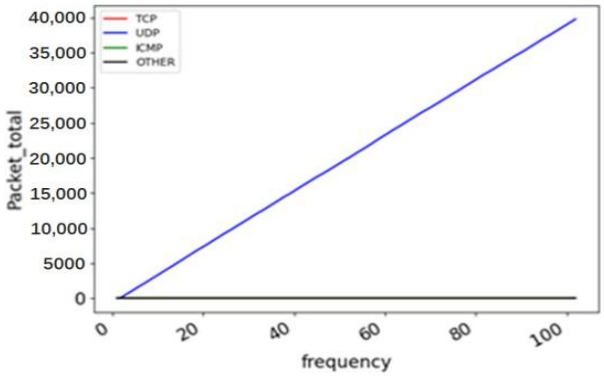
assault. The cumulative is the total number of packets, with the number of counts every 3 seconds as the abscissa.



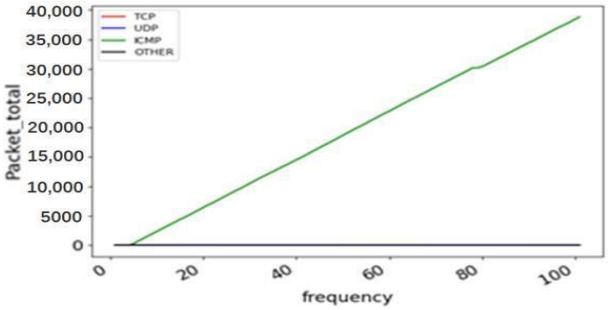
(a)



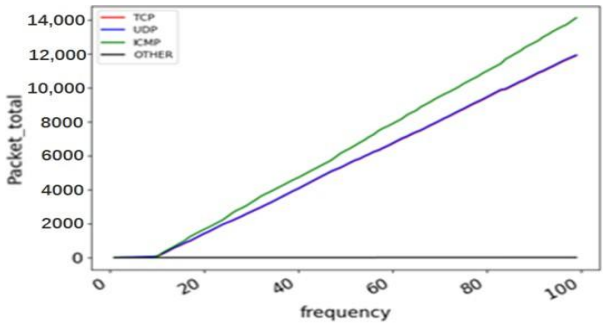
(b)



(c)



(d)



(e)

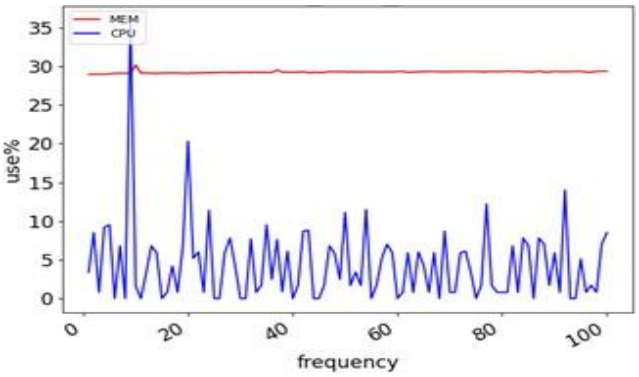
Figure 16. Packet flow cumulative diagram. (a) Normal transmission. (b) SYN flood attack; (c) UDP flood attack. (d) ICMP flood attack (e) MIX flood attack.

The total number of TCP, UDP, ICMP, and other packets during normal transmission, as well as during SYN flood attack, UDP flood attack, ICMP flood assault, and MIX flood assault and other situations are shown in Figure 16a-e. Figure 16a shows that there are several types of packets and that the total number of packets increases gradually during typical transmission conditions. No of the type of flood attack, the overall quantity of packets abruptly rises. Additionally, when under attack, there are various cumulative counts for various packet kinds. These characteristics can be used to determine the sort of attack as well as whether an attack has occurred. There are 100 recordings because the experimental retrieval time is 5 minutes. Figure 16b shows that after the SYN flood attack, IP1 had a significant rise in TCP packets, with a total of roughly 50,000 TCP packets passing through IP1 in under five minutes, while other types of packets are nearly equal to zero. Figure 16c of the UDP flood assault shows that UDP packets have significantly grown while other types of packets have nearly stabilized at zero. The 5 minute collection period. It is clear that throughout this time, there were a total of roughly 40000 UDP packets sent and received. Figure 16d of the ICMP flood assault shows that the amount of ICMP packets increased significantly, while the amount of other packet types decreased. The collecting time is five minutes, and the value is almost 0. The total number of ICMP packets likewise reached roughly 40,000 during this time. The circumstance of being assaulted by other packets (MIX

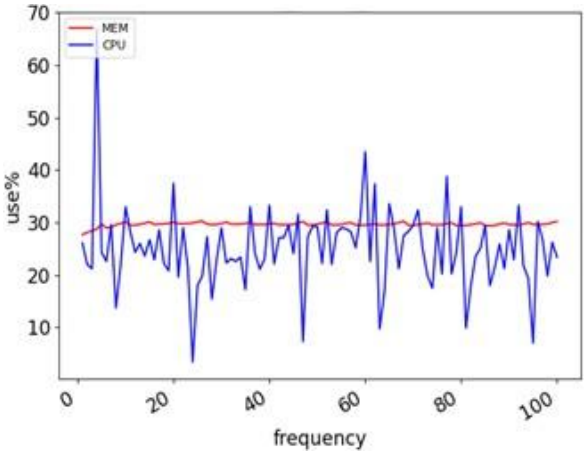
flood attack) is depicted in Figure 16e. The server is now processing a high number of different sorts of packets, with about equal numbers of packets having UDP and TCP characteristics. Therefore, in our experiment, it is clear to see that when attacked, various packets rise with this time and the amount of packets changes. The CPU and memory consumption rates are shown in Figure 17a-e, respectively. Figure 17a illustrates how little the CPU utilization rate varies during typical transmission, But during an attack, it will change significantly. In a similar vein, normal transmission uses significantly less memory than attackers do. It aids in determining if the system is being attacked. Figure 17b shows that the CPU utilization rate is around 20% greater during a SYN flood attack than it is under normal circumstances (Figure 17a). Figure 17c demonstrates how the CPU utilization rate swings wildly during UDP flood attacks, reaching a peak of more than 50%. Figure 17d shows how the CPU use rate changes drastically when the ICMP flood attack is encountered. The oscillation is in the similar situation as the UDP flood assault, but it differs in some ways. Figure 17e illustrates what happens when the MIX flood occurs. Like the preceding attacks, the attack will cause a rapid and dramatic increase in the CPU's utilization rate. Figure 17b-e demonstrates that the memory use rate is essentially unchanged from what is typical. The variation in CPU utilization allows us to deduce its properties. Table 4's results show how different attacks will impact the network speed during transmission. The typical network transmission speed in our experiment will decrease from an average of 14.2 Mbps to roughly 0.3 Mbps. Author deploy a neural network for edge computing and model training in light of the aforementioned experimental findings.

Table 4. Average speed under the normal transmission, SYN flood attack, UDP flood attack, ICMP flood attack, and MIX flood attack

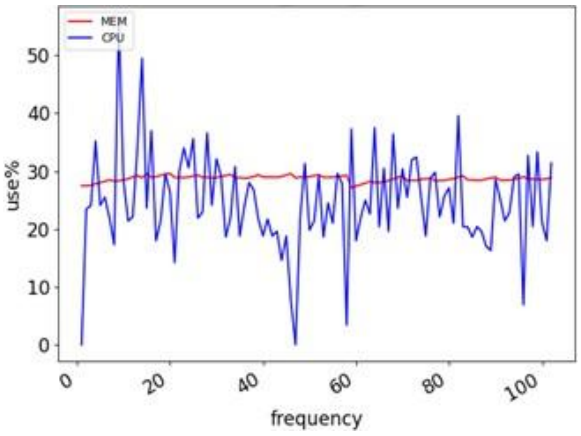
Condition	Value
Normal Transmission	14.2 Mbps
SYN Flood Attack	0.30 Mbps
UDP Flood Attack	0.30 Mbps
ICMP Flood Attack	0.31 Mbps
MIX Flood Attack	0.31 Mbps



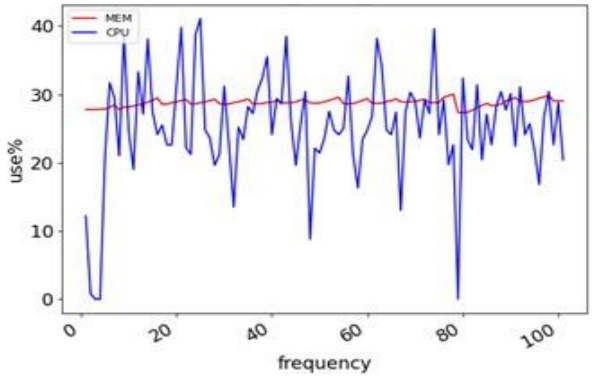
(a)



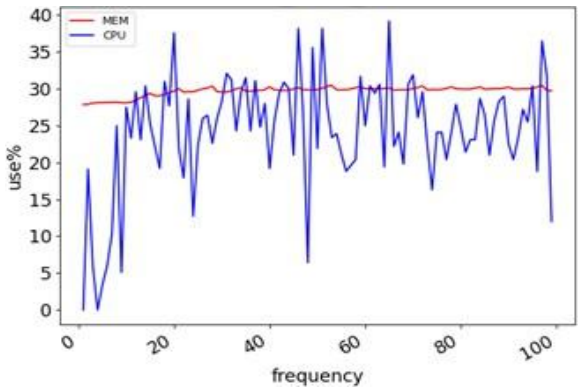
(b)



(C)



(d)



(e)

Figure 17. CPU and memory usages (a) normal transmission. (b) SYN flood attack; (c) UDP flood attack; (d) ICMP flood attack. (e) MIX flood attack.

This study trains and evaluates the models of a NN, a one-dimensional CNN, and a two-dimensional CNN. The practical validation of the suggested system will use the top-performing model. Accuracy = TP + TN/TP + TN + FP + FN*100% is a formula used to calculate the accuracy of predictions of true and false under all circumstances. Table 5 lists the definitions of the four conditions: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

Table 5. Definition of TP, FP, FN, and TN.

Transmission Mode	AI Model Identification Results	
	Normal Transmission	Abnormal Transmission
Normal	TP	FN
Abnormal	FP	TN

The three models stated above were trained using the packet traffic dataset in this study. The training settings are 10, 50, 100, 150, and 200 times, respectively, and there is just one data set utilized as the labelled input. Each training has a batch size of 2500 and a learning rate of 0.00001. The model has three hidden layers with 128, 64, and 32 neurons each, each with a "ReLU" excitation function, and uses the "softmax" excitation function for classification in the output layer. Four bits of data are utilized as label input and trained 10, 20, 30, 40, and 50 times each when the packet feature dataset is used for training. The learning rate is 0.00001, and the batch size is 2500. The excitation function is "ReLU," and the model comprises three hidden layers with 128, 64, and 32 neurons each, respectively. The output layer's categorization uses the "softmax" excitation function. In order to avoid over fitting, a dropout layer is inserted after the first layer for one-dimensional and two-dimensional CNN. Tables 6 and 7 display the respective training accuracies for the two distinct training parts. Tables 6 and 7 show that the accuracy of training can ultimately reach more than 99% with an increase in the quantity of training times and input ratio. This demonstrates that there must be more data collected than a specific level in accordance with the environmental characteristics. Tables 6 and 7 show that the accuracy of training can ultimately reach more than 99% with an increase in the quantity of training times and input ratio. This demonstrates that in order to achieve good training results, more data must be collected in accordance with the environmental conditions. The two-dimensional CNN model, also known as the traffic model and feature model, respectively, is more accurate after training in terms of both packet traffic and feature training, as shown in Tables 6 and 7. As a result, it will be used as the identification model for the practical validation.

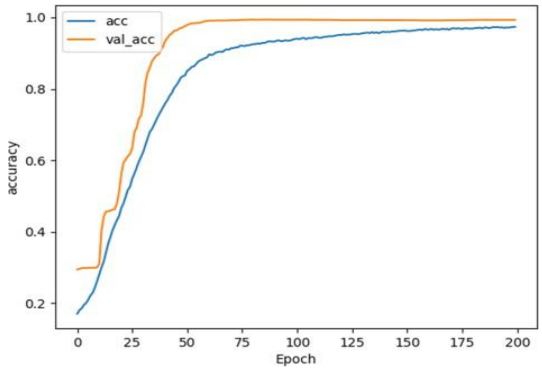
Table 6. Accuracy of NN, 1D CNN, and 2D CNN trained on a dataset of packet traffic.

Model	Number of Training				
	10	50	100	150	200
NN	49%	83.8%	99.5%	99.2%	99%
1D CNN	51%	53.5%	98.9%	99.3%	99%
2D CNN	15.6%	43.2%	99.4%	99.5%	99.8%

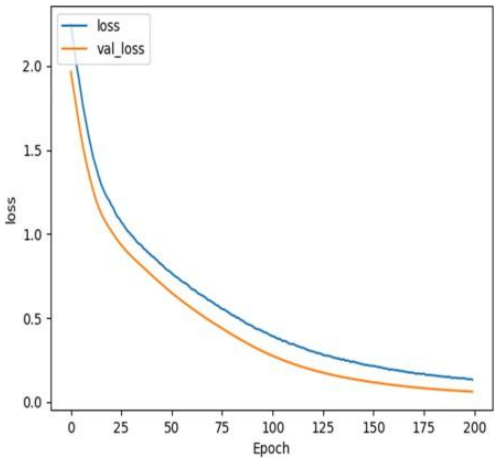
Table 7. Accuracy for NN, 1D CNN, and 2D CNN trained by using packet feature dataset.

Model	Number of Training				
	10	20	30	40	50
NN	58.5%	86.4%	98.8%	99.3%	99.7%
1D CNN	93.6%	95.7%	99.9%	99.8%	99.8%
2D CNN	97.5%	99.7%	99.8%	99.9%	99.9%

Figures 18 and 19 respectively display the two-dimensional CNN's accurate rate and loss rate for the traffic model and feature model. For real experiment verification in this study, the author used a trained two-dimensional CNN model. The amount and characteristics of packets passing through the data collection server IP1 are gathered under normal transmission, SYN flood attack, UDP flood attack, ICMP flood attack, and MIX flood assault, respectively. It will be transmitted to the edge computing computer through IP2 in order to use a trained model to determine the server's current state. The data will be pre-processed by the edge computing computer before being input into the traffic model and feature model, respectively, for identification. The definitive determination combining the collected identification rates and their weights yields the final identification result. As a result, this paper gives the traffic model and feature model weights of 70% and 30%, respectively, as the basis for judging the identification results, taking into consideration that the data of the packet traffic dataset may be impacted by the rising IoT user population and that the features captured will not be able to identify because the data captured is too small model and feature model the weights of 70% and 30% respectively.

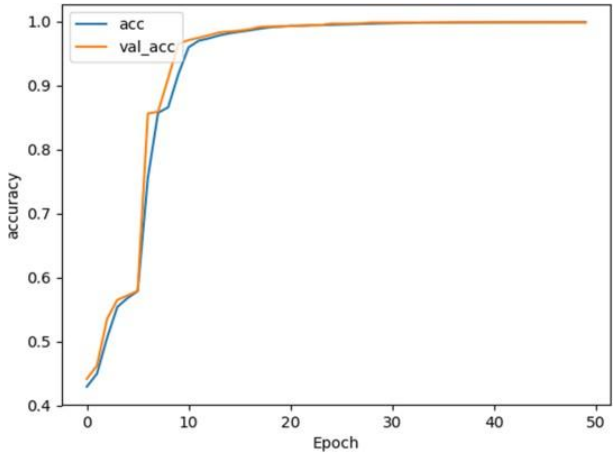


(a)

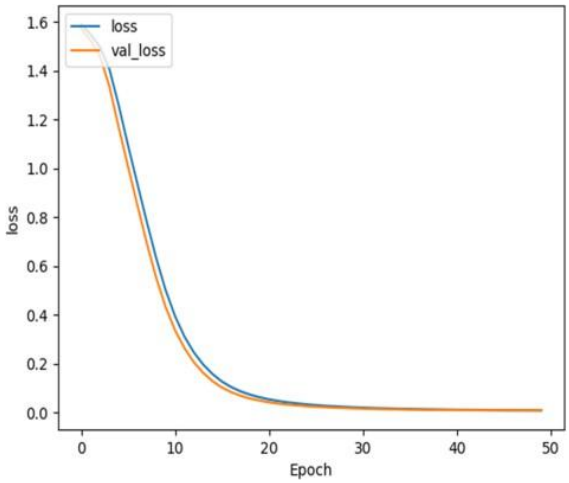


(b)

Figure 18: (**a**) Accuracy rate; (**b**) Loss rate of the two-dimensional CNN for the packet traffic model.



(a)



(b)

Figure 19: (a) Accuracy rate, (b) loss rate of the two-dimensional CNN for the packet feature model.

Table 8 lists the identification results on the edge computer measured under various conditions, where Feature denotes the identification result of the feature model, Flow denotes the identification result of the traffic model, and Weighted denotes the identification result following individual weighting on the two models. The proposed edge computing architecture with a trained CNN model can identify objects correctly under various transmission conditions, including normal transmission, SYN flood attack, UDP flood attack, ICMP flood attack, and MIX flood assault, according to the table. Although the traffic model can still recognize it when the data is uncommon, the feature model cannot, even in the absence of packets case during regular transmission. 70% of the weight is accounted for by the traffic model The result of the weighted identification is typical. The

main reason for the aforementioned experimental results is that this system uses two separate two-dimensional CNN models that were independently trained using the packet traffic dataset and the packet features dataset, respectively. It also simultaneously recognizes the system's current state and adds the appropriate weights to them as a basis for judgment. When packets are exchanged in the system, this design may make the most use of information about changes in linked features, effectively enhancing recognition accuracy and compensating for the drawbacks of utilizing a single model for judgment.

Table 8 shows the time interval identification for SYN flood attacks, UDP flood attacks, and ICMP.
flood Attack, and MIX flood Attack.

Condition	Feature	Flow	Weighted
Normal Transmission (Lack of packets)	Lack data	Normal	Normal
Normal Transmission	Normal	Normal	Normal
SYN Flood Attack	SYN Flood	SYN Flood	SYN Flood
UDP Flood Attack	UDP Flood	UDP Flood	UDP Flood
ICMP Flood Attack	ICMP Flood	ICMP Flood	ICMP Flood
MIX Flood Attack	MIX Flood	MIX Flood	MIX Flood

Table 9 displays the period of time—referred to as the identification time—between when a Raspberry Pi transmits the data it has captured to an edge computing device and when it receives the identification information it has returned. This table indicates that the system identification delay brought on by the DDoS attack is not related to the identification time.

Table 9. The identification time under the normal transmission, SYN flood Attack, UDP flood Attack, ICMP flood Attack, and MIX flood Attack.

Condition	Identification Time (s)
Normal Transmission	8.15
SYN Flood Attack	8.00
UDP Flood Attack	8.75
ICMP Flood Attack	8.20
MIX Flood Attack	8.23

5. Conclusion

This study suggests an edge computing-based DDoS assault detection solution. To lessen the impact of DDoS attacks on the data transmission in the IoT system, the edge computing computer in this system employs a trained two-dimensional CNN model to determine whether the data collection server in the IoT is currently under a DDoS attack and how the attack is being conducted. Due to the adoption of the edge computing architecture, the suggested DDoS detection system can be correctly constructed without altering the original IoT hardware structure. To better recognize DDoS attacks, two two-dimensional CNN models are utilized simultaneously. These models were trained using data from packet traffic and packet characteristics, respectively.

REFERENCES

1. Ziaei, M.; Zamani, B.; Bohlooli, A. A Model-Driven Approach for IoT-Based Monitoring Systems in Industry 4.0. In Proceedings of the 2020 4th International Conference on Smart City, Internet of Things and Applications (SCIOT), Mashhad, Iran, 16–17 September 2020; pp. 99–105.
2. Mirkovic, J.; Prier, G.; Reiher, P. Attacking DDoS at the source. In Proceedings of the 10th IEEE International Conference on
Network Protocols, Paris, France, 12–15 November 2002; pp. 312–321.
3. Wang, S.; Gomez, K.; Sithamparanathan, K.; Asghar, M.R.; Russello, G.; Zanna, P. Mitigating DDoS Attacks in SDN-Based IoT Networks Leveraging Secure Control and Data Plane Algorithm. Appl. Sci. 2021, 11, 929.
4. Lin, H.-C.; Wang, P.; Lin, W.-H.; Huang, Y.-H. A Multiple-Swarm Particle Swarm Optimisation Scheme for Tracing Packets Back to the Attack Sources of Botnet. Appl. Sci. 2021, 11, 1139.
5. Yan, Q.; Huang,W.; Luo, X.; Gong, Q. A multi-level DDoS mitigation framework for the industrial Internet of Things. IEEE Comm.
Mag. 2018, 56, 30–36.
6. Rodrigues, B.; Bocek, T.; Lareida, A. A blockchain-based architecture for collaborative DDoS mitigation with smart contracts. In Proceedings of the IFIP International Conference on Autonomous Infrastructure, Management and Security, Zurich, Switzerland,
10–14 July 2017; Springer: Cham, Switzerland, 2017; pp. 16–29.
7. Hinton, G.E.; Osindero, S.; Teh, Y. A Fast-Learning Algorithm for Deep Belief Nets. Neural Comput. 2006, 18, 1527–1554.
8. Doriguzzi-Corin, R.; Millar, S.; Scott-Hayward, S. Martínez-del-Rincón, J.; Siracusa, D. Lucid: A Practical, Lightweight Deep Learning Solution for DDoS Attack Detection. IEEE Trans. Netw. Serv. Manag. 2020, 17, 876–889.
9. Manimurugan, S.; Al-Mutairi, S.; Aborokbah, M.M. Chilamkurti, N.; Ganesan, S.; Patan, R. Effective Attack Detection in Internet of Medical Things Smart Environment Using a Deep Belief Neural Network. IEEE Access **2020**, 8, 77396–77404.

10. Roopak, M.; Tian, G.Y.; Chambers, J. Deep Learning Models for Cyber Security in IoT Networks. In Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 7–9 January 2019.pp. 0452–0457.
11. Alzahrani, R.J.; Alzahrani, A. Security Analysis of DDoS Attacks Using Machine Learning Algorithms in Networks Traffic. Electronics 2021, 10, 2919.
12. Ali, M.H.; Jaber, M.M.; Abd, S.K.; Rehman, A.; Awan, M.J.Damaševiˇcius, R.; Bahaj, S.A. Threat Analysis and Distributed Denial of Service (DDoS) Attack Recognition in the Internet of Things (IoT). Electronics **2022**, 11, 494.
13. Evmorfos, S.; Vlachodimitropoulos, G.; Bakalos, N. Gelenbe, E. Neural network architectures for the detection of SYN flood attacks in IoT systems. In Proceedings of the 13th International Conference on Pervasive Technologies Related to Assistive Environments, Corfu, Greece, 30 June–3 July 2020; pp. 1–4.
14. Asad, M.; Asim, M.; Javed, T.; Beg, M.O.; Mujtaba, H. Abbas, S. Deepdetect: Detection of distributed denial of service attacks using deep learning. Comput. J. 2020, 63, 983–994.
15. Sudharsan, B.; Sundaram, D.; Patel, P.; Breslin, J.G.; Ali, M.I. Edge2Guard: Botnet attacks detecting offline models for resource constrained IoT devices. In Proceedings of the 2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerComWorkshops), Kassel, Germany, 22–26 March 2021; pp. 680–685.
16. Jia, Y.; Zhong, F.; Alrawais, A.; Gong, B.; Cheng, X. FlowGuard: An intelligent edge defense mechanism against IoT DDoS attacks.IEEE Internet Things J. **2020**, 7, 9552–9562.
17. Su, J.; Vasconcellos, D.V.; Prasad, S.; Sgandurra, D. Feng, Y.; Sakurai, K. Lightweight classification of IoT malware based on image recognition. In Proceedings of the 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, Japan, 23–27 July 2018; Volume 2, pp. 664–669.
18. Jing, H.;Wang, J. Detection of DDoS Attack within Industrial IoT Devices Based on Clustering and Graph Structure Features.Secur. Commun. Netw. 2022, 9.
19. Vormayr, G.; Zseby, T.; Fabini, J. Botnet Communication Patterns. IEEE Commun. Surv. Tutor. 2017, 19, 2768–2796.
20. Najafimehr, M.; Zarifzadeh, S.; Mostafavi, S. A Hybrid Machine Learning Approach for Detecting Unprecedented DDoS Attacks.J. Supercomput. 2022, 78, 8106–8136.
21. Ravi, N.; Shalinie, S.M. Learning-Driven Detection and Mitigation of DDoS Attack in IoT via SDN-Cloud Architecture. IEEE Internet Things J. 2020, 7, 3559–3570.
22. Sahi, A.; Lai, D.; Li, Y.; Diykh, M. An Efficient DDoS TCP Flood Attack Detection and Prevention System in a Cloud Environment.IEEE Access 2017, 5, 6036–6048.