www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

Two-Fold Machine Learning Approach to Prevent and Detect IoT Botnet Attacks

Vanapamula Veerabrahmachari, Shaik Guntur Mahabub Subhani, Arekatla Madhava Reddy,

Dr. Padigala Suresh

[3] Assistant Professor, [1,2,4] Associate Professor

vveerabrahmachari@gmail.com, subhanimehandi@gmail.com

amreddy2008@gmail.com, padigalas36@gmail.com

Department of CSE, A.M. Reddy Memorial College of Engineering and Technology, Petlurivaripalem,

Narasaraopet, Andhra Pradesh -522601

ABSTRACT

The most common kind of cyber attack in an IoT setting, botnet attacks consist of many stages that culminate in a distributed denial of service (DDoS) assault once scanning has begun. The majority of the current research is devoted to identifying botnet assaults after hacked IoT devices have begun launching DDoS attacks. Not only that, but the majority of botnet detection algorithms that use machine learning can only work with a certain dataset. Due to the variety of attack patterns, these methods do not perform well on different datasets. Hence, we generate 33 different kinds of scans and 60 different kinds of DDoS assaults to provide a generic scanning and DDoS attack dataset in this study. To ensure that the machine learning algorithms were trained with the most comprehensive attack coverage possible, we also partly combined the scan and DDoS attack samples from three publicly-available datasets. We then suggest a two-pronged machine learning strategy for detecting and preventing botnet assaults on the Internet of Things. To stop IoT botnet assaults in their tracks, we created a top-notch deep learning model—ResNet-18—to identify scanning activities during the early stages of an attack. Also, in order to identify botnet assaults on the Internet of Things, we trained a second ResNet-18 model for distributed denial of service detection. When it comes to detecting and preventing IoT botnet assaults, the suggested two-pronged method generally achieves 98.89% accuracy, 99.01% precision, 98.74% recall, and 98.87% f1-score. We evaluated the performance of three additional ResNet-18 models trained on three other datasets for identifying scan and DDoS assaults to that of the proposed two-fold technique to show how effective it is. The experimental findings demonstrate that the suggested dual strategy outperforms previous trained models in botnet attack prevention and detection.

IoT botnet, Internet of Things, botnet detection, botnet assaults, botnet distributed denial of service, index terms

I. INTRODUCTION

New data shows that the proliferation of vulnerable IoT devices is directly correlated with the rising frequency of cyberattacks [6]. The most common types of cyberattacks, which have grown in both frequency and severity over the last decade, are botnet and distributed denial of service (DDoS) assaults [4], [6]. When launching a botnet assault, cybercriminals first search a network for Internet of Things (IoT) devices that aren't well protected. The hacker then uses the analyzed data to identify susceptible Internet of Things (IoT) devices in order to implant malware using a bot software [7]. Once installed, bot software links infected devices to a central server or peer network. From there, commands are sent to carry out various malicious activities, such as flooding a target server or website with spam or DDoS attacks [6, 8], etc. Attackers launch distributed denial of service (DDoS) assaults using compromised Internet of Things (IoT) devices when they join a botnet. Insecure internet of things (IoT) gadgets aren't the only ones botnet attacks pose a major risk to [6]. Internet of Things (IoT) botnet assaults have been steadily rising since the Mirai attack in 2016 [9]. The Mirai botnet has given rise to several versions and imitators since its source code was made public [9]. Over the last several years, these new variations and imitators have infected millions of IoT devices and caused increasingly huge and catastrophic distributed denial of service (DDoS) assaults, such as those on GitHub [10], Amazon Web Services [11], etc.

Internet of Things (IoT) vulnerabilities may be quickly and readily discovered by cybercriminals using tools like Shodan [12], Censys [13], etc. Insecure Internet of Things devices may be targeted using the massive amounts of data provided by these internet search engine providers [9]. Once an attacker obtains access to unsecured IoT devices, they may launch a barrage of cyberattacks, including phishing, spamming, distributed denial of service (DDoS) [6, 8, 9], and many more. Due to the fact that hijacked IoT devices conduct a broad variety of DDoS

assaults, many recent studies have shown that these devices are particularly vulnerable to botnet and DDoS attacks [14], [15]. Also, according to Gartner's latest prediction, unsecured IoT devices are responsible for 25% of cyberattacks [16]. An effective security solution to identify IoT bots is necessary to prevent unsecured IoT devices from becoming bots and launching various DDoS assaults. There are two main types of methods now used to identify botnet and DDoS attacks: those that rely on hosts and those that rely on networks [17]. Internet of Things (IoT) devices do not have the resources necessary to implement host-based solutions [1], [17]. This is because IoT devices have limited memory, battery life, and processing power. Internet of Things (IoT) devices and networks are vulnerable to severe cyberattacks, but a network-based solution offers superior protection. There are primarily three categories into which the network-based approaches fall.One approach is the signature-based detection technique, which uses a database of predefined criteria to compare incoming network traffic with known malicious actors.
2) An anomaly-based approach to detection establishes a baseline profile for every device interacting in the network by analyzing its typical behavior. A significant departure from the norm is regarded as an abnormality. Two other categories of anomaly-based detection methods includeSection V: Detection based on statistics: These techniques use a distribution of incursions to identify abnormalities.
Detects anomalies using packet and payload information; based on machine learning. These techniques rely heavily on machine learning models for attack detection and prevention.
_ A knowledge-based detection approach uses a network's profile or historical data to spot suspicious activity. To identify network anomalies, many test scenarios are used to establish the network's profile, or prior knowledge [22].
Thirdly, there's the specification-based detection approach, which identifies intrusions according to user-defined specifications or criteria [22]. The signature-based detection system has one big flaw: it can only identify known threats that have rules stored in its database [20], [21]. Contrarily, tasteful protocol-based detection algorithms can only decipher encrypted traffic to a limited extent. Nevertheless, there is a very successful method for analyzing encrypted traffic and identifying unknown threats called traffic behavior analysis, also known as anomaly detection [19]. In the realm of anomaly detection technologies, the machine learning approach has recently shown remarkable performance. The detection algorithms that rely on machine learning are taught to discriminate between normal and attack traffic by analyzing datasets [20], [21]. In the future, machine learning models may identify new bonnet and DDoS attacks that are variations or imitators of current ones by learning the normal and attack traf c patterns.
After infecting Internet of Things devices with malware and directing them to carry out criminal operations, current botnet attack detection technologies identify the botnet. In addition, the majority of current botnet detection techniques that rely on machine learning can only work with the datasets that were used for training [6]. Because various datasets include various kinds of botnet assaults, this is the case. As a result of the variety of botnet assaults, the traits that were effective in identifying them in one dataset are insufficient for detecting them in another [6]. Consequently, many methods struggle to handle diverse attack patterns on various datasets [6]. Internet of Things (IoT) devices must be protected against botnet and distributed denial of service (DDoS) assaults even in their early stages, while scanning is taking place, if they are to avoid compromise. Consequently, we provide a unique two-pronged strategy to stop botnet attacks in their early stages (i.e., scanning attacks) and to identify distributed denial of service (DDoS) attacks in internet of things (IoT) networks in the event that an attacker gains access to an IoT device and begins to launch a DDoS assault. We have already shown that an attacker may employ bot-infected IoT devices to carry out many harmful actions, such as flooding DDoS assaults[6,8], sending spam emails, etc., however our main emphasis here is on identifying such attacks. A state-of-the-art deep learning model, ResNet, is used in the proposed two-pronged method. ResNet is taught to identify scanning activity and subsequently to detect DDoS attacks conducted by attackers or hacked IoT devices either within or outside the network.
First, we trained the ResNet-18 [23] model for scanning attack detection to protect the Internet of Things (IoT) devices and network from IoT botnet attacks. This model can identify the early stage of an attack and alert us to the malicious attempt before the attacker goes further with compromising the IoT devices. Second, we trained the ResNet-18 [23] model for DDoS attack detection to identify and counteract botnet attacks, should an intruder compromise the scanning attack detection model, infect Internet of Things devices with malware, and launch DDoS assaults.

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

Notation	Description
ANN	Artificial Neural Network
C&C	Command and Control
CNN	Convolution Neural Network
DCNN	Distributed CNN
DDoS	Distributed Denial of Service
DNN	Deep Neural Network
FN	False Negative
FP	False Positive
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
IoT	Internet of Things
LOF	Local Outlier Factor
LR	Logistic Regression
MLP	Multi-Layer Perceptron
PSI	Printing String Information
P2P	Peer-to-Peer
RNN	Recurrent Neural Network
ResNetDDoS-1	Resnet-18 Model Trained over DDoSLab Dataset
ResNetDDoS-2	Resnet-18 Model Trained over DDos Samples of CICIDS-19 Dataset
ResNetDDoS-3	Resnet-18 Model Trained over DDos Samples of CICIDS-17 Dataset
ResNetDDoS-4	Resnet-18 Model Trained over DDos Samples of Bot-IoT Dataset
ResNetScan-1	Resnet-18 Model Trained over ScanLab Dataset
ResNetScan-2	Resnet-18 Model Trained over Scan Samples of CICIDS-19 Dataset
ResNetScan-3	Resnet-18 Model Trained over Scan Samples of CICIDS-17 Dataset
ResNetScan-4	Resnet-18 Model Trained over Scan Samples of Bot-IoT Dataset
SDN	Software-Defined Network
SGD	Stochastic Gradient Descent
TCP	Transmission Control Protocol
TN	True Negative
TP	True Positive
UDP	User Datagram Protocol

II. RELATED WORK

Several methods have been suggested for detecting botnet attacks thus far. You may classify the current botnet detection methods into two main categories: graph-based and flow-based [4]. By comparing the network's communication patterns to those of nearby nodes, graph-based botnet detection algorithms may identify suspicious activity [24]. Contrarily, _ow-based botnet identification uses machine learning algorithms to keep an eye on incoming and outgoing traf_c data, or features, and identify botnet assaults according to how similar their patterns are. Using printed string information (PSI) graphs, Nguyen et al. [16] suggested a graph-based method for detecting the IoT botnet. In order to train a convolutional neural network (CNN), a deep learning model, to identify IoT botnets, the authors first used PSI graphs to extract high-level characteristics from the function call graph. For their part, Wang et al. [24] put out BotMark, an autonomous model. They suggest a strategy that uses a combination of flow-based and graph-based network traffic analysis to identify botnet assaults. Using stability and similarity scores between _ows, k-means performs the _ow-based detection. In contrast, graph-based detection analyzes anomaly scores using the local outlier factor (LOF) and the least-square approach. A similar approach was put out by Yassin et al. [25] in their innovative method. It combines many techniques, including using the frequency process on registry data, visualizing graphs, and generating rules. Utilizing a graph-theoretical framework, the writers examined the Mirai assaults. The writers used directed graphs to find Mirai patterns that were similar and those that were different. The planned strategy is entirely focused on the Mirai assault.

To identify newly deployed botnets, Almutairi et al. [27] presented a hybrid method that uses host level detection, network level detection, and a mix of the two. With a particular emphasis on DNS botnet traf_c, the writers zeroed down on HTTP, P2P, and IRC. There are three parts to the suggested method: the host analyzer, the network analyzer, and the detection report. Naïve Bayes and a decision tree were the two machine learning methods used by the writers for traffic classification. A bot identification approach called BotFP, short for bot _ngerprinting, was also suggested by Blaise et al. [28]. The first version of the BotFP framework, BotFP-Clus, uses clustering methods to group instances of similar traffic, while the second version, BotFP-ML, uses two supervised ML algorithms, SVM and MLP, to learn from signatures and detect new bots. Similarly, a strategy for detecting IoT botnet attacks based on machine learning was created by Soe et al. [30].

A model builder and an assault detector are the two steps that make up the suggested model. Model builder is a step-by-step process that includes data gathering, data classification, model training, and feature selection. Similar to the model builder phase, the attack detector stage begins with packet decoding and feature extraction. In order to identify botnet attacks, the characteristics are then sent to the attack detector engine, where models such as artificial neural networks (ANNs), J48 decision trees, and Naïve Bayes are used. To identify botnet attacks on the Internet of Things, Sriram et al. [31] presented a system based on deep learning. The suggested method for detecting IoT botnet

attacks takes network traffic flows into account; these flows are transformed into feature records before being fed into the deep neural network (DNN) model.
In a pair of trials, Nugraha et al. [32] tested four different deep learning models to see how well they could identify botnet attacks. When it came to detecting botnet assaults, the testing findings showed that CNN-LSTM performed better than any deep learning model. A cloud-based, distributed deep learning framework was suggested by Parra et al. [33]. Phishing and Internet of Things botnet assaults may be detected by their framework. Two machine learning models make up their model. One is a distributed convolutional neural network (DCNN) that can identify URL-based assaults on a client's Internet of Things (IoT) devices. The other is a combination of recurrent neural networks (RNNs) and long short-term memory (LSTM) networks that can detect Botnet attacks on the backend. Network _ow traf_c was used by Pektacs and Acarman [34] to do botnet detection using deep learning. The goal of deploying the suggested deep neural network was to determine whether the traf_c was malicious or not. There is an effort to improve the model's performance by exploring buried layers and neurons. A 99% success rate has been attained using the suggested model.
Similar to this, Ahmed et al. [35] suggested a deep learning approach for detecting botnet assaults. The use of deep learning for botnet attack detection on software-defined networks (SDNs) was suggested by Maeda et al. [36]. The authors assessed the detection accuracy of the deep learning model for botnet identification by training it using data gathered on ow-based traffic from the botnets. One deep learning model that the scientists employed to identify infected IoT devices was a multi-layer perceptron (MLP). A new method for detecting botnet attacks on the Internet of Things (IoT) using deep auto-encoder was also developed by Meidan et al. [18]. The nefarious traf_c network infected nine Internet of Things devices using the popular Mirai and BASHLITE botnets. The authors trained deep auto encoders independently on benign and attack traffic for each IoT device.
A hybrid two-stage intrusion detection system (IDS) was suggested for the Internet of Things (IoT) environment by Bovenzi et al. [37]. They propose a two-step process for identifying and categorizing network traffic abnormalities into different types of attacks. The first stage involves detecting anomalies. For the first step of anomalies detection, the authors utilized a multi-modal deep auto encoder. For the second stage, they used three machine learning classifiers. To identify abnormalities on local network traffic by using an unsupervised learning technique, Mirsky et al. [39] have suggested a plug-and-play network intrusion detection system (IDS) called Kitsune, which makes use of auto-encoders. The authors tested the system in both online and offline modes using a dataset they created themselves for botnet attacks. Their suggested method outperformed state-of-the-art anomaly detectors.
Table 2 provides a concise overview of the works mentioned before, outlining their distinguishing features. It is clear that the majority of the current methods for detecting botnet assaults relied on traffic-based machine learning techniques.
The variety of attack patterns also means that most of these methods don't work well on different datasets [6].

TABLE 2. A review of some existing botnet detection techniques.

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

Ref.	Dataset	Approach	Technique	Attack Type	Pre-attack/ Post-attack
Nguyen *et al.* [16]	IoT-PoT [26]	Graph-based	PSI graphs, CNN	IoT botnet	Post-attack
Wang *et al.* [24]	Self-collected	Hybrid	LOF, K-means	Botnet	Post-attack
Yassin *et al.* [25]	Self-collected	Graph-based	Directed graphs	Mirai attack	Post-attack
Almutairi *et al.* [27]	Self-collected	Flow-based	NB, DT	HTTP, P2P, IRC, DNS	Post-attack
Blaise *et al.* [28]	CTU-13 [29]	Flow-based	LR, RF, SVM, MLP	Botnet	Post-attack
Soe *et al.* [30]	N-BaIoT [18]	Flow-based	NN, DT, NB	Mirai , BASHLITE	Both
Sriram *et al.* [31]	N-BaIoT [18]	Flow-based	DNN	Mirai , BASHLITE	Both
Nugraha *et al.* [32]	CTU-13 [29]	Flow-based	CNN-LSTM, CNN, LSTM, MLP	Botnet	Post-attack
Parra *et al.* [33]	N-BaIoT [18]	Flow-based	DCNN, RNN, LSTM	Mirai , BASHLITE	Both
Pektacs *et al.* [34]	CTU-13 [29]	Hybrid	NN	Botnet	Post-attack
Ahmed *et al.* [35]	CTU-13 [29]	Flow-based	NN	Botnet	Post-attack
Maeda *et al.* [36]	CTU-13 [29]	Flow-based	MLP	Botnet	Post-attack
Meidan *et al.* [18]	N-BaIoT [18]	Flow-based	DAE	IoT botnets, Mirai, BASHLITE	Post-attack
Bovenzi *et al.* [37]	Bot-IoT [38]	Flow-based	M2-AE, RF, NB, MLP	IoT Botnet	Both
Mirsky *et al.* [39]	Self-collected	Flow-based	AE	IoT Botnet	Both

Where PSI: Printing String Information, CNN: Convolution Neural Network, LOF: Local Outlier Factor, NB: Naive Bayes, DT: Decision Tree, LR: Logistic Regression, RF: Random Forest, SVM: Support Vector Machine, MLP: Multi-layer Perceptron, NN: Neaural Network, LSTM: Long-short Term Memory, DCNN: Distributed CNN, RNN: Recurrent Neural Network, AE: Auto-encoder, DAE: Deep AE, M2-AE: Multi-modal AE

III. PRELIMINARIES

A. COMPONENTS OF AN IoT BOTNET

There are typically four parts to a botnet. All of these parts work together, and they comprise the bot software, zombie gadget, bot-master, and C&C server.
The bot-master, who may be a human attacker or a computer program, is the first node in a botnet. Internet of Things (IoT) devices that are being targeted are scanned by the bot-master. The bot-master then uses the scan findings to instal a bot software on the susceptible IoT devices, taking advantage of their vulnerabilities. Bots may be programmed to carry out destructive actions by connecting to a command and control (C&C) server or bot-master.
Sections that follow provide a high-level overview of each part:

1) Robot Machine
An attacker may install bot programs on compromised devices as malware. The bot software infects the victim's IoT device and communicates with the command and control server or bot-master to carry out harmful operations, such as flooding assaults, spamming, etc. [6, 8].

2) ZOMBIE PROPHET
The attacker installs a bot software on a victim's physical device, turning it into a zombie device. These gadgets include smart cameras, smart televisions, smart wearables, and so on in the context of the Internet of Things.
3. BOT-MASTER
Botnets are led by a bot-master or bot header.
In order to coordinate botnet assaults, it might be a hacker or a computer software that the hacker controls. In client-server or peer-to-peer botnet architectures, the bot-master is the primary operator who gives orders to the command and control server or to specific bots.
four. C&C Server The zombie devices are controlled by the central computer known as the C&C server, which receives control signals from the bot-master. Not all botnets need a command and control server. In a P2P network, the botmaster communicates with the zombies directly.
B. An Internet of Things Botnet's Life:

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

According to [9], the botnet assault consists of many stages. There are five steps for an insecure Internet of Things device to transform into a bot capable of distributing spam, launching distributed denial of service attacks, etc. Sometimes, these steps are called the botnet life cycle. As seen in Figure 1, these phases consist of scanning, malware insertion, botnet connection, command execution, maintenance, and augmentation. At the beginning of a botnet's life cycle, the attacker performs scanning to gather information for later use. An attacker may use the data gathered during scanning to introduce malware into the device by exploiting a vulnerability.
Afterwards, the malware that was injected connects to the bot-master and follows its orders. Lastly, the bot-master keeps the infected devices running well and updates them when necessary so the virus may be passed on. The sections that follow provide detailed descriptions of each of these steps:

1) SCANNING

FIGURE 1. Five stages of an IoT botnet attack life cycle.

The life cycle of a botnet begins with scanning. At this point, the assailant or bot-master gathers basic information about the target device's services, protocols, operating system, etc. by scanning the target network or device. In order to scan an IoT network or device, the attackers employ many tactics. Scanning technologies such as NMap [40], ZMap [41], Masscan [42], OpenVAS [43], etc. are widely utilized.

2) Injection via Malware

The attacker will use several exploitation techniques to find the vulnerabilities in the target network or device after gathering information about it. An attacker may inject malware onto a target device via effective exploitation. Malware sent via phishing, email attachments, etc., is another way an attacker might lure a victim in addition to exploiting vulnerabilities.
The perpetrator is able to go on with the assault when the victim falls for the phishing website or opens an attachment in their inbox without realizing it is malicious software. A shellcode is first installed on the victim's device by the attacker. An early infection is another name for this stage. While the shellcode is running, it gathers further information about the victim's device and transmits it to a central server. From there, a bot program binary and some extra configurations are downloaded. Installing the bot software with the target device characteristics is the next step [44], [45]. A secondary injection is another name for this procedure. By installing the bot application, the victim's device turns into a "zombie" [45].

3) HOSTNET LINK

Upon startup, the bot software connects to the bot-master or command and control server in order to be recognized as a legitimate bot. Rally refers to the zombie's first efforts to connect to the command and control server in order to get more directives from the bot-master [45]. After malware is implanted, it begins to carry out the attacker's plan. In order to trigger the malware's execution, the attacker might program it to function as a Trojan horse. Malware establishes a connection between the infected system and the bot-master, where it gets instructions on what to do next.

4) Executing a Command

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

When a compromised device establishes communication with a command and control server, it joins the ranks of a botnet army. The bot-master then instructs it to carry out nefarious operations, and it waits for directions from the command and control server. The waiting phase is another name for this stage. Scanning for new bots, transmitting spam, and undertaking denial-of-service attacks are all examples of harmful operations [6, 8, 46, etc.]. A bot-master may launch a flooding or distributed denial of service assault against a target server or network when it detects that a large number of infected devices are linked to it.

5) Improvements and Upkeep

In order for a bot software to stay undetected on a victim PC, it has to be updated and maintained over time. This is a critical step for an attacker to keep the malware infection going, so they may exploit the infected system for further assaults and criminal operations. Both conventional and Internet of Things botnets have very similar life cycles [5]. Distinctions are limited to intended devices. The goal of typical botnet attacks is to compromise PCs, servers, and other similar devices; however, with IoT botnet attacks, the objective is Internet of Things devices such as smart cameras, smart TVs, etc. [5].

IV. PROPOSED METHODOLOGY

To identify and stop botnet assaults in Internet of Things (IoT) networks, we presented a new two-pronged machine learning strategy in this paper. The first step in protecting the IoT network against botnet assaults was training a state-of-the-art deep learning model, ResNet-18 [23], to identify the (pre-attack stage) scanning activities. Second, we trained a different ResNet-18 [23] model to identify DDoS attacks launched by malicious actors after they have compromised poorly protected Internet of Things devices.

We have established that there are five steps to a botnet attack's lifecycle: scanning, malware insertion, botnet connection, command execution, and maintenance and upgrades. The botnet assault begins with scanning, which is the first step of an early attack. In order to prevent an attacker from moving on to next phases of an attack, the suggested approach halts them mid-scanning activity. Thus, the suggested technique finds botnet assaults by recognizing the DDoS attack for incoming and outgoing traf_c, and it stops botnet attacks by detecting scanning attack activity.

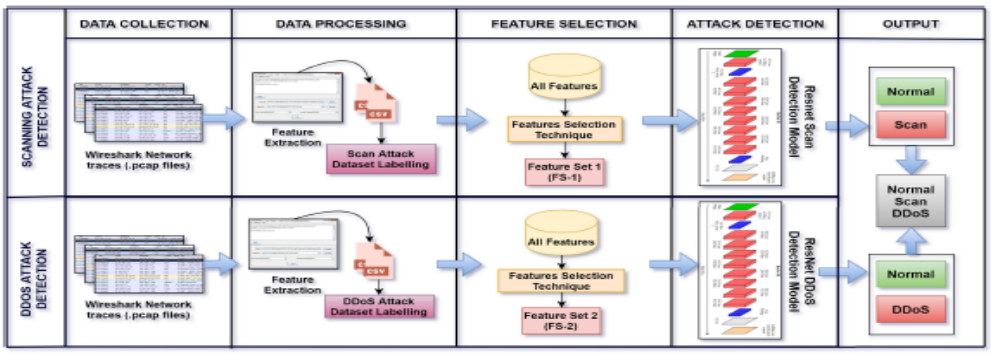


Figure 2 shows the five main steps that each fold of the suggested method takes to scan for and identify DDoS assaults. To begin training the machine learning models, we first produced and recorded the scanning and DDoS attack traffic in.pcap format. We retrieved the features and saved them in.csv files after converting the network packet traces (.pcap files) into _ows in the second step. In addition, we annotated the dataset with terms like "normal" to show benign traf_c, "scan" to indicate scanning traf_c, and "DDoS" to indicate DDoS assault traffic.

To maximize the machine learning model's performance with the fewest possible unique features, we used the Logistic Regression (LR) feature selection method in the third stage.

The LR feature selection approach was used because of its effective performance in previous studies [6, 8, 17]. Additionally, when compared with other feature selection methods, it is quick, easy, and complex-free [6, 8, 17]. Finally, we trained a ResNet-18 [23] model for scanning and another for DDoS attack detection using the generated feature vector. The taught machine learning models are then put to the test to ensure they can withstand real-time

assault scenarios. We have established that scanning is the starting point and the DDoS assault is the finishing touch to an Internet of Things botnet attack. Hence, the suggested architecture is composed of a pair of machine learning models: one to identify and stop DDoS assaults and the other to stop IoT botnet attacks. To stop IoT botnet attacks before they start, during the scanning stage, we trained the ResNet-18 [23] model on the scanning assaults dataset in addition to the standard traf_c dataset. This will protect the IoT devices and network against botnet attacks while they are still in the early stages. We named this model ResNetScan-1 because it is able to identify scanning assaults conducted by attackers during the early stages of an attack in order to gather information about susceptible IoT devices, therefore preventing IoT botnet attacks. Conversely, if a hacker gains access to the scanning attack detection mechanism, they may spread malware across IoT devices and launch distributed denial of service (DDoS) assaults. Afterwards, we trained an additional ResNet-18 [23] model using the DDoS attack and regular traffic datasets to identify DDoS attack activity carried out by infected industrial IoT devices. A ResNetDDoS-1 model was the name we gave to this model.

In order to categorize the incoming network data as scan, DDoS, or normal, we combined the ResNetScan-1 and ResNetDDoS-1 models, as seen in Figure 2. The procedures for developing and testing the suggested machine learning models to identify and thwart IoT botnet assaults are detailed in the sections that follow.

Part A: Detecting Scanning Attacks

In the first round, we used the five procedures outlined earlier to train a ResNet-18 [23] model for scanning attack detection. The sections that follow will outline these procedures.

First, gather data.

The suggested process for scanning attack detection begins with data collecting. To begin, we had a look at the methods currently used by cybercriminals to search IoT networks and devices for sensitive data [47]_[51].

From the literature research [47]_[51], we culled eleven distinct scanning techniques that attackers often use to learn about the susceptible IoT in the early stages of an attack. The following scanning methods are executed using three popular scanning tools: Nmap [40], Hping3 [52], and Dmitry [53]: SYNACK scan, FIN-ACK scan, NULL scan, UDP scan, TCP Window scan, TCP connect scan, and banner capturing. We used three distinct scanning techniques—horizontal scan, vertical scan, and box scan—to conduct these scanning assaults on two lab servers and produce the scan traf_c for this study. In order to create and gather scanning traffic, a total of eleven hundred thirty-three different kinds of scanning assaults were executed. On an Ubuntu system with a Core i7 CPU and 8 GB RAM, we installed three popular scanning programs: Nmap [40], Hping3 [52], and Dmitry [53]. We then conducted scanning attacks using these tools. We then create a few python scripts to run various scanning instructions. As part of the scanning assaults, we used the Wire shark program to capture the network packets and saved them as. cap files [54].This dataset was created by ScanLab itself.

We used three publicly-available datasets—CICIDS-19[55], CICIDS-17[56], and Bot-IoT[38]—to supplement the network packets we generated and collected from our experimental setup. In order to assess the performance of ResNet-18 [23] models trained over different datasets for scanning attack detection, we collect the scan and normal samples of these three publicly-available datasets. The models were trained using the ScanLab dataset.

2) Pre-processing Data

We must pre-process the data once we have captured the scanning traffic. Using the Cyclometer [57] Tool, we retrieved the features from the captured. cap files of the ScanLab dataset as part of the pre-processing stage. Using a ve-tuple that includes the source IP, destination IP, source port, destination port, and protocol, the CICFlowmeter [57] utility scans a given. cap file and extracts over sixty flow characteristics for each flow that is identified. At [58], you may get more information on these characteristics that were derived by Cyclometer [57]. An output file named "a.csv _le" containing the flow characteristics of a specific PCAP file is produced by the Cyclometer [57]. The data that comes out of it is not tagged. We categorized the network traffic as "normal" and "scan" according to the IP addresses that were utilized for scanning.

In a similar vein, we labeled the result.csv files according to the descriptions of the CICIDS-19 [55], CICIDS-17 [56], and Bot-IoT [38] datasets after preprocessing. To improve the training of machine learning algorithms, we eventually combined the scan attack samples from the ScanLab dataset with the 50K samples randomly selected from these three datasets. This allowed us to achieve maximum attack coverage.

3. Choosing Features

After we have extracted the characteristics from all.pcap files, the next step is to choose the most relevant ones so that a machine learning model can differentiate between regular and scan traffic. The LR technique was used for features selection since it has shown greater performance in previous research projects [6], [8]. First, we used the LR method to choose the top 20 features from four datasets: ScanLab, two others, and one more. using that, we conducted a frequency analysis of the features chosen by the LR algorithm from the ScanLab dataset and all three datasets, using the procedure outlined in [6]. Feature set 1 (FS-1) consists of the fifteen characteristics that were chosen most often by the LR algorithm; these features are reported in Table 3 and shown in Figure 2. In order to use them in the succeeding phases of scanning attack detection, the fifteen attributes specified in Table 3 are chosen from each dataset.

4) An ML Model for Smart Scan Detection Training

Following the feature selection process, we divided each dataset into three parts: the train set, the validation set, and the test set. To achieve this goal of efficiently training the ML model and avoiding overfitting, we randomly picked 60% of the data for training, 20% for validation, and 20% for testing. During the training phase, both the training set and the validation set are used. The machine learning model is trained using the training set. At the end of each training session, we test the trained model on the validation set; this information is used by the optimizer method to change the ML model's weights, making the training process more efficient. The last step is to put the ML model through its paces on a collection of unseen data, or test set, once training is complete. We trained the ResNet-18 [23] model using the ScanLab dataset's train set, as previously stated.
Image processing and computer vision challenges [4] involving high-dimensional arrays of pictures were initially the target of the ResNet-18 [23] model's classification efforts. As the ResNet-18 [23] model is known to over_t at low dimensional input [4], [59], we must transform the data into high dimensional arrays before to beginning the training process. Thus, we began by transforming the whole ScanLab dataset into 15 _ 15 _ 1 high dimensional arrays and then stored them as pictures according to the procedure outlined in [4]. The remaining three datasets were also transformed into 15 _ 15 _ 1 greyscale pictures. Not to mention the hyperparameters—learning rate, batch size, epoch count, and optimizer—that must be mentioned as well. The learning rate determines how often the weights of the ML model are updated according to the estimated error at the end of each epoch. The epochs provide information on the total number of iterations used to train a machine learning model. In order to facilitate efficient and rapid training, the batch size partitions the provided dataset into smaller portions.
A machine learning model's speed and performance may be enhanced by having the optimizer select the weights of a neural network with the best features. Therefore, the learning rate is determined.

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

TABLE 3. Top 15 features selected for scanning and DDoS attack Detection using LR algorithm.

Top 15 Features (FS-1) for Scanning Attack Detection	Top 15 Features (FS-2) for DDoS Attack Detection
Total Backward Packets	Flow Duration
Fwd Packet Length Min	Total Length of Fwd Packets
Fwd Packet Length Mean	Total Length of Bwd Packets
Fwd Packet Length Std	Total Fwd Packets
Bwd Packet Length Min	Total Backward Packets
Bwd Packet Length Mean	Fwd Packet Length Mean
Bwd Packet Length Std	Bwd Packet Length Mean
Bwd Header Length	Flow Bytes/s
Min Packet Length	Flow Packets/s
Max Packet Length	Fwd IAT Mean
Packet Length Mean	Bwd IAT Mean
Down/Up Ratio	Fwd Packets/s
Avg Packet Size	Bwd Packets/s
Avg Fwd Segment Size	Packet Length Mean
Avg Bwd Segment Size	Down/Up Ratio

TABLE 4. TCP DDoS attack patterns with different flags combination.

1 Flag Patterns	2 Flags Patterns	3 Flags Patterns	4 Flags Patterns	5 Flags Patterns
SYN (S)	SP, SR, SA, SF, SU	SPU, SFU, SFP, SFR, SRP, SRU, SPA, SFA, SAU	SFRA, SFRU, SFRP, SFAU, SFPA, SFPU, SRPU	SFRPA, SFRAU, SFRPU, SFPAU, SRPAU
ACK (A)	AR, AU, AP	AFP, AFR, ARU, APU, AFU	AFRP, ARPU, AFPU, AFRU	AFRPU
RST (R)	RF, RU, RP	RPU, RFU, RFP	RFPU	
URG (U)	UP, UF	UFP		
FIN (F)	FA, FP			
PSH (P)				

5) Verification and Testing

We put the trained ResNet-18 [23] model through its paces on the test set when training is complete. Given that the trained model does not have access to the test set, we used four widely-used performance indicators to assess the trained model's efficacy on the test set. Section V explains these performance measures. In Section V, you can also see the results of the trained model's performance on the test set for detecting DDoS attacks.
In four stages, we assessed the forecasts of all the trained models to verify the efficacy of the suggested technique. Starting with the CICIDS-19[55], CICIDS-17[56], and Bot-IoT [38] datasets, which were not used for training the proposed ResNetDDoS-1 model, we conduct tests on these datasets. The performance of additional saved models was also cross-validated on the dataset that was not used during training. At last, we checked how each ResNetDDoS model fared on every dataset.

V. RESULTS AND DISCUSSION

A. EXPERIMENTAL SETUP

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

Previous discussion indicated that the primary objective of this research was to examine preexisting methods of scanning and DDoS assaults. In light of the findings, we used three separate network traffic generating tools—Nmap[40], Hping3[52], and Dmitry [53]—to produce thirty-three distinct kinds of scanning assaults and sixty different kinds of distributed denial of service (DDoS) attacks. This whole suite of utilities was set up on an Ubuntu-18 operating system-powered Core i7 PC with 8 GB of RAM. The network traffic that was produced was recorded in.pcap format using the Wireshark utility [54]. Then, we labelled the workstations in the experiment based on their IP addresses and retrieved characteristics from these.pcap files.
We next used feature selection approaches and partitioned the dataset into a train set and a test set so that we could go forward with the suggested methodology stages by steps as outlined in.

TABLE 5. ResNet-18 model architecture.

Layer	Output Size	Kernel
conv1	112 x 112	7 x 7, 64, stride 2
conv2	56 x 56	3 x 3 max pool, stride 2 $\begin{bmatrix} 3\times 3, & 64 \\ 3\times 3, & 64 \end{bmatrix} \times 2$
conv3	28 x 28	$\begin{bmatrix} 3\times 3, & 128 \\ 3\times 3, & 128 \end{bmatrix} \times 2$
conv4	14 x 14	$\begin{bmatrix} 3\times 3, & 256 \\ 3\times 3, & 256 \end{bmatrix} \times 2$
conv5	7 x 7	$\begin{bmatrix} 3\times 3, & 512 \\ 3\times 3, & 512 \end{bmatrix} \times 2$
prediction	1 x 1	average pool, fc, softmax

Chapter Four. The suggested two-pronged method may identify two types of cyberattacks: scan assaults and distributed denial of service attacks. These two assaults couldn't be more unlike. Similarly, as shown in Table 3, the top traits that emerged from the feature selection process varied for the two assaults. Hence, to improve the detection and prevention of IoT botnet assaults, we trained two ResNet-18 [23] models independently. There are a total of 18 layers in the ResNet-18 [23], with 10 of those layers serving as convolutional and 8 as pooling. The design of the ResNet-18 [23] model used in this study is shown in Table 5. There were a total of 11,185,666 computational parameters in our ResNet-18 [23] model; 11,176,066 of these parameters were trainable, whereas 9600 were not. We utilized Python 3 and the TensorFlow v2.2 package in a Google Colab environment to train and test the ResNet-18 [23] model for this study.
In Table 6, you can see how much time was spent training and testing the two ResNet-18 [23] models in the suggested two-pronged method. You can see from Table 6 that the ResNetDDoS-1 model takes (49.45/63668 D) 776.685 _s to test one picture, while the ResNetScan-1 model takes (5.75/12546 D) 458.313 _s. Thus, compared to utilizing only one detection model, using a second model resulted in 1.59 times more latency (when treating the ResNetDDoS-1 as a single main model).

Section B: Evaluating Performance

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

We employ four widely-used performance metrics to assess the suggested two-pronged strategy for identifying and deterring IoT Botnet assaults. The precision, recall, accuracy, and F1-score are some of the factors that are considered. Below, these parameters are defined. First, we define the following terms so that we may compute these performance parameters:Chapter V: The TP: It was an assault that the ML model correctly identified.

TABLE 6. Time constraints of the proposed Two-fold approach.

Model	Dataset	Execution Time
ResNetDDoS-1	Training Images: 1,91,006	1884 sec
	Testing Images: 63,668	49.45 sec
ResNetScan-1	Training Images: 50,184	368 sec
	Testing Images: 12,546	5.75 sec

_ **True Negative (TN):** The normal _ow was accurately predicted as normal by the ML model.
The ML model incorrectly identified a normal flow as an attack, resulting in a false positive (FP).
The ML model incorrectly identified the is attack flow as normal, which results in a false negative (FN).
No. 1: Precision The ratio of attack flows properly classified as "attack flows" (TP) to regular traffic flows correctly classified as "normal flows" (TN) is the definition. The mathematical definition of it is (1):

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \times 100 \qquad (1)$$

2) **Precision:** It tells about how many of the predicted attack _ows were correct. Mathematically, it is described as (2):

$$Precision = \frac{TP}{TP + FP} \times 100 \qquad (2)$$

3) **Recall** - It de_nes the ability of the system to correctly detect the attack upon the occurrence of the actual attack. It is also called as sensitivity. Mathematically, it is expressed as (3):

$$Recall = \frac{TP}{TP + FN} \times 100 \qquad (3)$$

4) **F1-Score** - It is de_ned as the weighted harmonic mean of precision and recall. Mathematically, it is de_ned as (4):

$$F1{-}Score = 2 \times \frac{Recall * Precision}{Recall + Precision} \qquad (4)$$

1) TEST SCENARIO 1: WHEN EACH ResNetScan MODEL IS TRAINED AND TESTED OVER SIMILAR DATASET
As a whole, the ResNetScan models' performance on the test-set of the traf_c, a comparable dataset, where they were trained to identify scanning attacks, as summarized in Table 7. Table 7, first row, displays the results of the ResNetScan-1 model's training on the train-set taken from the ScanLab dataset using the ResNet-18 [23] model. For the purpose of categorizing normal and scanning attack traf_c, the ResNetScan-1 model achieved 99.20% accuracy, 99.39% precision, 99.05% recall, and 99.22% f1-score when evaluated on the same dataset (i.e., the ScanLab dataset) that it was trained on.

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

TABLE 7. Testing results when each ResNetScan model is trained and tested over the similar dataset for scanning attack detection.

Model Name	Trained & Test Over	Accuracy	Precision	Recall	F1-Score
ResNetScan-1	ScanLab	99.20	99.39	99.05	99.22
ResNetScan-2	CICIDS-19 [55]	99.91	100	99.83	99.92
ResNetScan-3	CICIDS-17 [56]	99.79	99.95	99.67	99.81
ResNetScan-4	Bot-IoT [38]	98.85	98.95	98.66	98.81

TABLE 8. Testing results when each ResNetScan model cross validated for scanning attack detection.

Testing Model	Testing Dataset	Accuracy	Precision	Recall	F1-Score
ResNetScan-1	CICIDS-19 [55]	99.91	100.00	99.83	99.92
	CICIDS-17 [56]	99.79	99.81	99.81	99.81
	Bot-IoT [38]	98.48	98.20	98.67	98.43
	Average Results	99.39	99.34	99.44	99.39
ResNetScan-2	ScanLab	66.75	74.51	53.79	62.47
	CICIDS-17 [56]	58.81	66.92	51.04	57.91
	Bot-IoT [38]	37.67	0.66	0.19	0.30
	Average Results	54.41	47.36	35.01	40.23
ResNetScan-3	ScanLab	94.24	99.81	33.63	50.31
	CICIDS-19 [55]	47.05	50.00	0.08	0.17
	Bot-IoT [38]	51.62	75.00	0.29	0.57
	Average Results	64.30	74.94	11.33	17.02
ResNetScan-4	ScanLab	56.53	89.95	17.47	29.26
	CICIDS-19 [55]	45.26	2.40	0.08	0.16
	CICIDS-17 [56]	44.46	40	0.09	0.19
	Average Results	48.75	44.12	5.88	9.87

Training the ResNet-18 [23] model on the CICIDS-19 [55] dataset also yields the ResNetScan-2 model. For identifying normal and scanning attack traf_c, ResNetScan-2 demonstrated 99.91% accuracy, 100% precision, 99.83% recall, and 99.92% f1-score when evaluated across the testset of its appropriate dataset, i.e., CICIDS-19 [55] dataset. As an example, Table 7 displays the results of several ResNetScan models on their respective datasets for normal and scanning traf_c detection. When trained and evaluated for scanning attack detection, all of the ResNetScan models demonstrated excellent performance with f1-scores, recall, accuracy, and precision above 98%. In terms of accurately identifying scan traf c and normal traf c, the ResNetScan-2 model was the most successful when tested on the test-set of the corresponding training dataset.

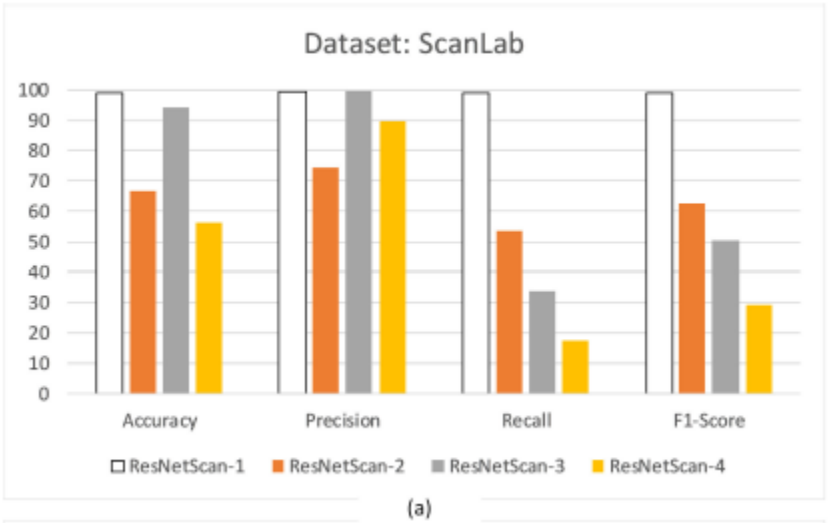
2) The Second Test Scenario: Comparing ResNetScan Models on Different Datasets

Here we ran tests to see how well each ResNetScan model fared when cross-validated against the test-set of different datasets. The results of all trials that evaluate each ResNetScan model for normal and scan traf_c detection are shown in Table 8. As a first step, we trained the ResNetScan-1 model on the ScanLab dataset and then applied it to the test-set of the CICIDS-19[55], CICIDS-17[56], and Bot-IoT[38] datasets. This model was derived from ResNet-18 [23]. The experimental findings show that ResNetScan-1 model did a good job of properly distinguishing normal and scan traf_c across all three datasets. When trained and evaluated on a comparable dataset, the ResNetScan-1 model achieved an average accuracy of 99.39%, precision of 99.34%, recall of 99.44%, and f1-score of 99.39%.

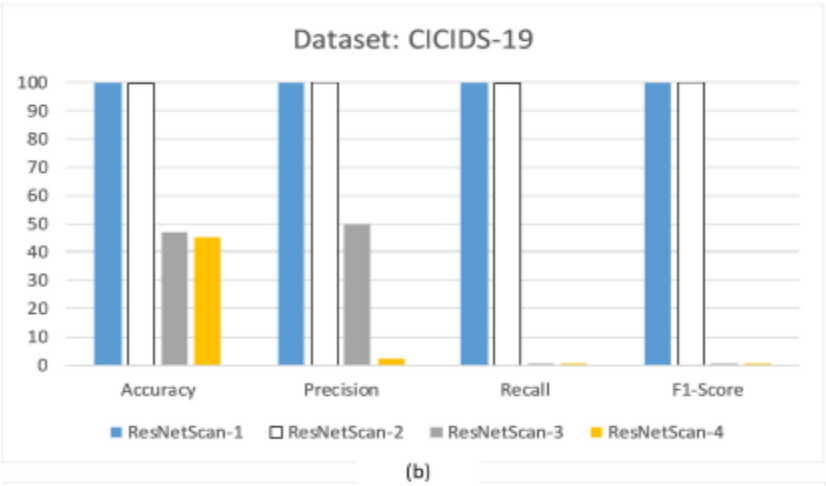
Likewise, we evaluated the ResNetScan-2 model's efficacy on the test-sets of ScanLab, CICIDS-17 [56], and Bot-IoT [38] in the second experiment. Table 8 displays the average experimental results for the ResNetScan-2 model,

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

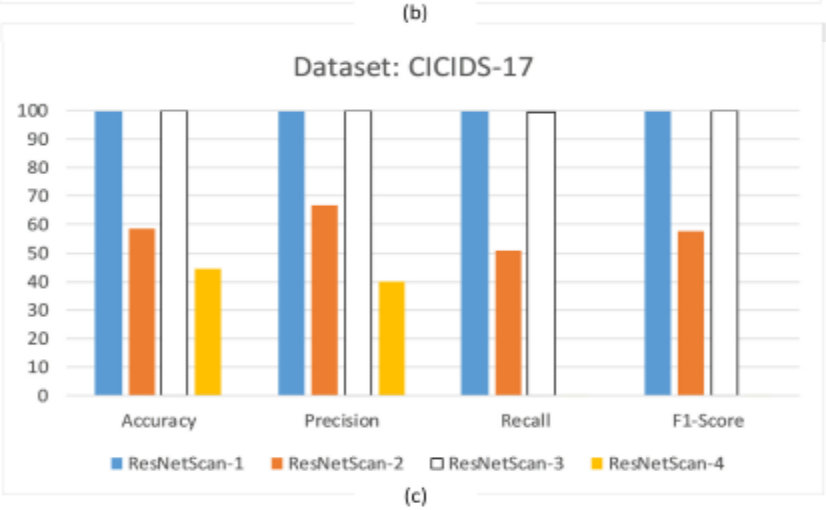
which were 54.41% accuracy, 47.36% precision, 35.01% recall, and 40.23% f1-score. When compared to the other scan datasets, the overall performance of the ResNetScan-2 model dropped by 45.50 percent, 52.64 percent,



(a)



(b)



(c)

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

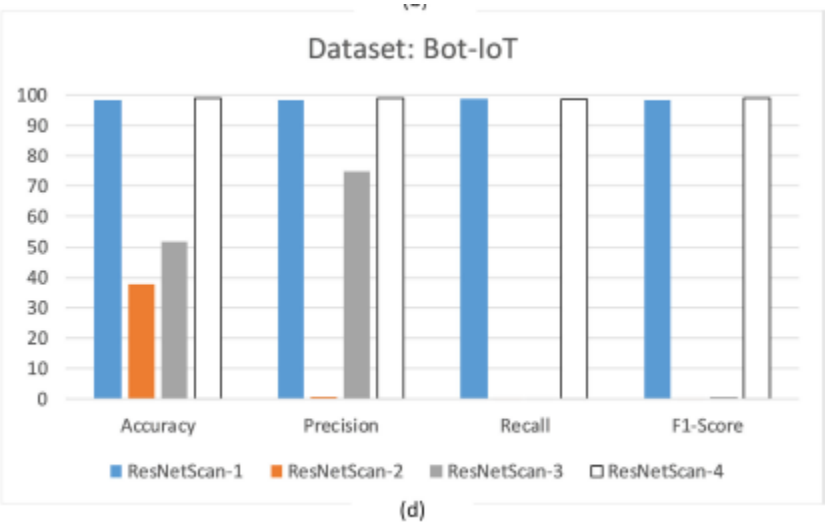


FIGURE 3. Performance comparison of ResNetScan models on individual dataset. (a) Presents the results with the ScanLab dataset. (b) Presents the results with the CICIDS-19 dataset. (c) Presents the results with the CICIDS-17. (d) Presents the results with Bot-IoT dataset.

64.82% for accuracy, 59.59% for precision, and 60.99% for recall and f1-score, respectively. So, to sum up, the ResNetScan-2 model's performance was much worse when tested and trained on a different dataset compared to the original. Similarly, we evaluated the ResNetScan-3 model's efficacy on the test-sets of ScanLab, CICIDS-19 [55], and Bot-IoT [38] in the third experiment. Results from the experiments show that the ResNetScan-3 model achieved an average of 64.30 percent accuracy, 74.9 percent precision, 11.3 percent recall, and 17.02 percent f1-score. Based on these findings, ResNetScan-3 efficiently identified typical traf_c samples, but it failed to accurately identify scan samples, leading to a significant drop in average f1-score compared to average precision.
Finally, in the fourth experiment, we used the scanLab test-set, CICIDS-17 [56] and CICIDS-19 [55] datasets to evaluate the ResNetScan-4 model's performance. Table 8 summarizes the experimental findings, which show that the ResNetScan-4 model achieved an accuracy of 48.75%, a precision of 44.12%, a recall of 5.88%, and a f1-score of 9.87%. The average f1-score terri_cally fell relative to the average precision, which is a consequence of the ResNetScan-4 model not adequately detecting the scan attack samples. All ResNetScan models' performance is compared across datasets in Fig. 3 (a)-(d). In Figure 3 (a)-(d), the white bars with the black border line show the results of the ResNetScan model that was trained and tested on the same dataset, while the other bars show the results of the different ResNetScan models on each dataset. Figure 3 (a)–(d) shows that when a ResNetScan model is evaluated on the test-set of the same or a comparable dataset, it performs the best.
Nevertheless, if we take a closer look at Figure 3 (a)-(d), we can see that the suggested ResNetScan model—specifically, the ResNetScan-1 model—achieved the second-best performance scores and surpassed all other ResNetScan models with outstanding accuracy, precision, recall, and f1-score. The experimental findings show that when evaluated on the dataset that was not used for training, the proposed ResNetScan-1 model performed better than all previous ResNetScan models. It is possible to identify an intruder in the second step if they breach the scan detection stage, hack an IoT device, and begin executing the DDoS assault. In addition, the DDoS attack detection model will identify an incoming DDoS assault and prevent the attacker from further harming the network in the event that an Internet of Things device or network becomes the target of a DDoS attack (i.e., the device is under attack from outside the network).

3) TEST SCENARIO 3: WHEN EACH ResNetDDoS MODEL IS TRAINED AND TESTED OVER SIMILAR DATASET

For this situation, we ran tests to see how well different ResNet-DDoS models performed when evaluated on the same dataset as their training. After being evaluated on the same dataset that was used for training, Table 9 displays the results of all ResNetDDoS models in identifying normal and DDoS attack traffic. Table 9's first row shows the

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

ResNetDDoS-1 model's performance. This model was trained using the train-set retrieved from the DDoSLab dataset, which is acquired from the ResNet-18 [23] model. Results for normal and DDoS assault traf_c classification using the ResNetDDoS-1 model on a comparable dataset (the DDoSLab dataset) showed an accuracy of 98.57%, a precision of 98.63%, a recall of 98.42%, and a f1-score of 98.52%.

Training the ResNet-18 [23] model on the CICIDS-19 [55] dataset also yields the ResNetDDoS-2 model. while examined on the test-set of its corresponding dataset, namely the CICIDS-19 [55] dataset, ResNetDDoS-2 demonstrated a f1-score of 99.52%, an accuracy of 99.49%, a precision of 99.40%, and recall of 99.63% while recognizing normal and scanning attack traf_c. Table 9 also shows the results of different ResNetDDoS models on their own datasets for identifying normal and DDoS traffic. It is clear that all of the ResNetDDoS models achieved respectable results in terms of recall, accuracy, precision, and f1-score.

In terms of accurately identifying both normal and DDoS traf_c, the ResNetDDoS-2 model proved to be the most effective when tested on the test-set of the corresponding training dataset.

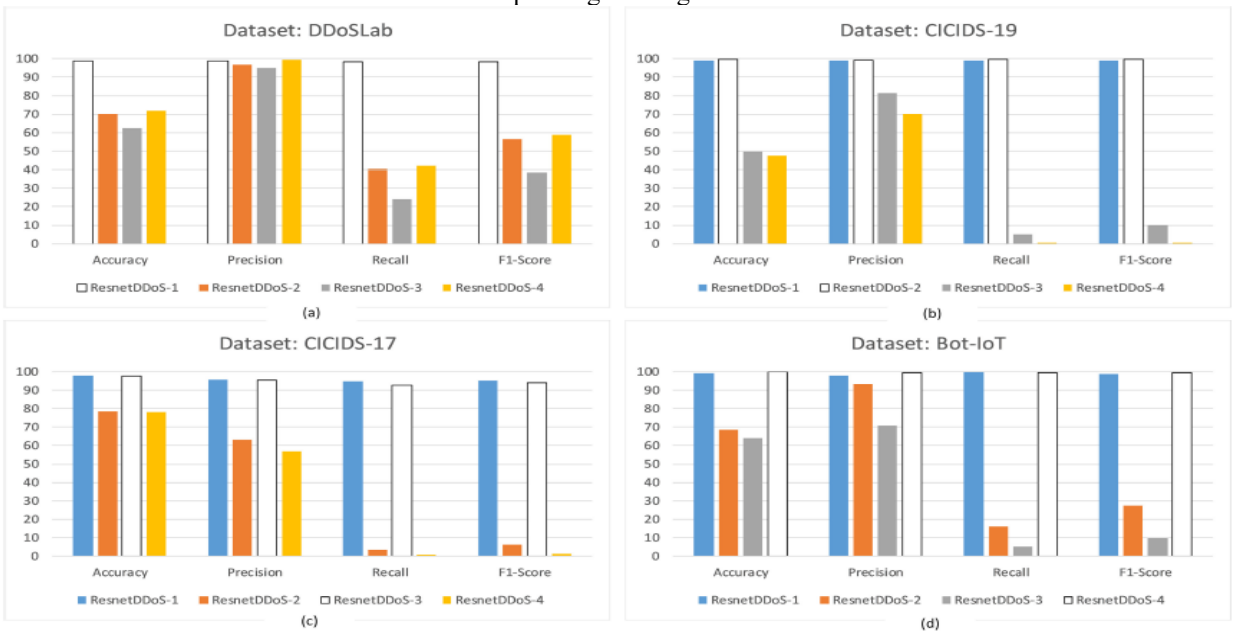


FIGURE 4. Performance comparison of ResNetDDoS models on individual dataset. (a) Presents the results with the ScanLab dataset. (b) Presents the results with the CICIDS-19 dataset. (c) Presents the results with the CICIDS-17. (d) Presents the results with Bot-IoT dataset.

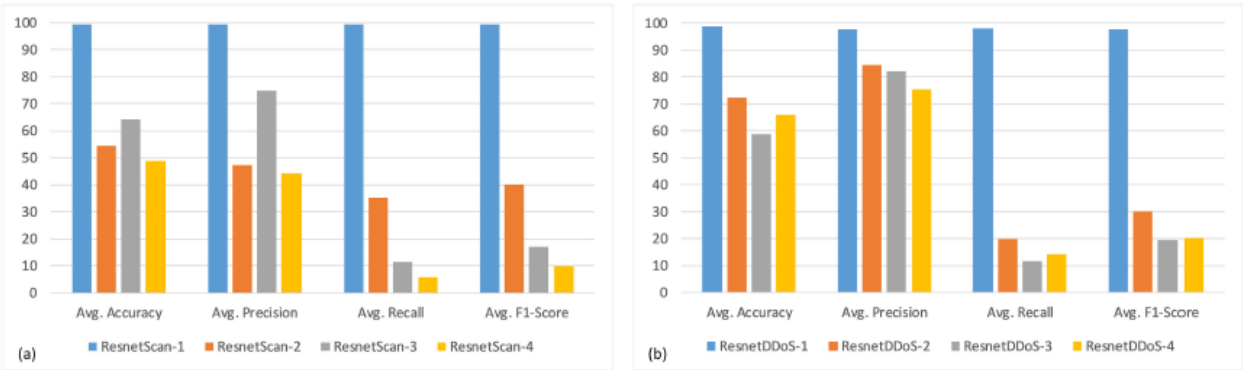


FIGURE 5. Average performance comparison of all (a) ResNetScan and (b) ResNetDDoS models when tested over other datasets that were not used while training.

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

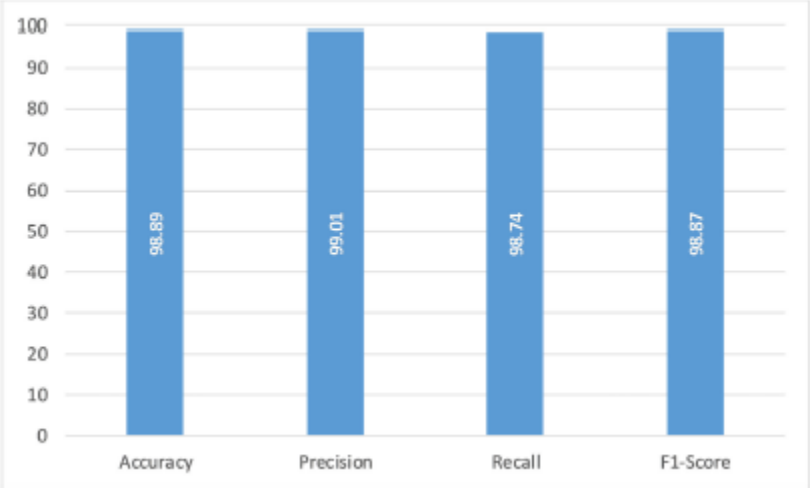


FIGURE 6. Overall performance of the proposed Two-Fold machine learning approach to prevent and detect IoT botnet attacks.

VI. CONCLUSION

To identify and stop botnet assaults on the Internet of Things, we presented a two-pronged machine learning strategy in this paper. As a first step, we built a top-tier deep learning model—ResNetScan-1—using ResNet-18 to identify scanning attacks. In the event that the scanning detection model is unable to avert a botnet attack, we trained a second ResNet-18 model, called the ResNetDDoS-1 model, to identify the DDoS assault. In a couple of experiments, we trained the ResNet-18 model on three publicly-available datasets using scan and DDoS traf_c samples. Then, we saved the resulting ResNetScan and ResNetDDoS models to verify the performance of the proposed ResNetScan-1 and ResNetDDoS-1 models. We then put each ResNetScan and ResNetDDoS model through its paces on the testset of different datasets, ensuring they were not trained on any of them. When evaluated on datasets that were not used for training, the performance of all ResNetScan and ResNetDDoS models—with the exception of the suggested ResNetScan-1 and ResNetDDoS-1 models—significantly decreased, according to the experimental findings. The trial findings also showed that the suggested ResNetScan-1 and ResNetDDoS-1 models continued to perform better than the competition when it came to identifying scan and DDoS assaults. So, to avoid and identify IoT botnet assaults with a wide range of attack patterns, the suggested dual strategy is effective and reliable.

So yet, our research has only covered 33 distinct scanning methods and 60 distinct DDoS assaults. We want to expand our coverage of scanning and distributed denial of service (DDoS) attack methodologies in the future so that the suggested framework may be trained to identify and block IoT botnet and DDoS assaults more effectively. Additionally, we may use the suggested dual strategy in an intrusion detection system to assess how well it performs on real-time network traffic.

REFERENCES

[1] I. Ali, A. I. A. Ahmed, A. Almogren, M. A. Raza, S. A. Shah, A. Khan, and A. Gani, ``Systematic literature review on IoT-based botnet attack," IEEE Access, vol. 8, pp. 212220_212232, 2020.

[2] S. Ghazanfar, F. Hussain, A. U. Rehman, U. U. Fayyaz, F. Shahzad, and G. A. Shah, ``IoT-Flock: An open-source framework for IoT traf_c generation," in Proc. Int. Conf. Emerg. Trends Smart Technol. (ICETST), Mar. 2020, pp. 1_6.

[3] M. Safaei Pour, A. Mangino, K. Friday, M. Rathbun, E. Bou-Harb, F. Iqbal, S. Samtani, J. Crichigno, and N. Ghani, ``On data-driven curation, learning, and analysis for inferring evolving Internet-of-Things (IoT) botnets in the wild," Comput. Secur., vol. 91, Apr. 2020, Art. no. 101707.

[4] F. Hussain, S. G. Abbas, M. Husnain, U. U. Fayyaz, F. Shahzad, and G. A. Shah, ``IoT DoS and DDoS attack detection using ResNet," in Proc. IEEE 23rd Int. Multitopic Conf. (INMIC), Nov. 2020, pp. 1_6.
[5] S. Dange and M. Chatterjee, ``IoT botnet: The largest threat to the IoT network," in Data Communication and Networks. Singapore: Springer, 2020, pp. 137_157.
[6] F. Hussain, S. G. Abbas, U. U. Fayyaz, G. A. Shah, A. Toqeer, and A. Ali, ``Towards a universal features set for IoT botnet attacks detection," in Proc. IEEE 23rd Int. Multitopic Conf. (INMIC), Nov. 2020, pp. 1_6.
[7] A. O. Proko_ev, Y. S. Smirnova, and V. A. Surov, ``A method to detect Internet of Things botnets," in Proc. IEEE Conf. Russian Young Res. Electr. Electron. Eng. (EIConRus), Jan. 2018, pp. 105_108.
[8] B. K. Dedeturk and B. Akay, ``Spam _ltering using a logistic regression model trained by an arti_cial bee colony algorithm," Appl. Soft Comput., vol. 91, Jun. 2020, Art. no. 106229.
[9] N. Vlajic and D. Zhou, ``IoT as a land of opportunity for DDoS hackers," Computer, vol. 51, no. 7, pp. 26_34, 2018.
[10] GitHub Survived Biggest DDoS Attack Ever Recorded. Accessed: May 3, 2021. [Online]. Available: https://github.blog/2018-03-01-ddosincident- report/