Received October 28, 2021, accepted November 21, 2021, date of publication November 25, 2021,
date of current version December 17, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3131014

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

FAISAL HUSSAIN[1], SYED GHAZANFAR ABBAS[1], IVAN MIGUEL PIRES[2,3], SABEEHA TANVEER[1], UBAID U. FAYYAZ[1], NUNO M. GARCIA[2], GHALIB A. SHAH[1], AND FARRUKH SHAHZAD[1]

[1]Al-Khawarizmi Institute of Computer Science (KICS), University of Engineering and Technology (UET), Lahore 54890, Pakistan
[2]Instituto de Telecomunicações, Universidade da Beira Interior, 6200-001 Covilhã, Portugal
[3]Escola de Ciências e Tecnologia, University of Trás-os-Montes and Alto Douro, 5001-801 Vila Real, Portugal

Corresponding authors: Faisal Hussain (faisal.hussain.engr@gmail.com) and Ivan Miguel Pires (impires@it.ubi.pt)

This work was supported in part by the Centro de Competências em Cloud Computing (C4) under Grant Operação Centro-01-0145-FEDER-000019, in part by the Programa Operacional Regional do Centro (CENTRO 2020) through the Sistema de Apoio à Investigação Científica e Tecnológica—Programas Integrados de IC&DT, in part by FCT/MEC through national funds, in part by FEDER—PT2020 Partnership Agreement under Project UIDB/50008/2020, and in part by the Al-Khwarizmi Institute of Computer Science (KICS), University of Engineering and Technology Lahore (UET), Lahore, Pakistan. This article is based upon work from COST Action IC1303–AAPELE–Architectures, Algorithms, and Protocols for Enhanced Living Environments and COST Action CA16226–SHELD-ON–Indoor living space improvement: Smart Habitat for the Elderly, supported by COST (European Cooperation in Science and Technology). COST is a funding agency for research and innovation networks. Our Actions help connect research initiatives across Europe and enable scientists to grow their ideas by sharing them with their peers. It boosts their research, career, and innovation. More information in www.cost.eu.

ABSTRACT The botnet attack is a multi-stage and the most prevalent cyber-attack in the Internet of Things (IoT) environment that initiates with scanning activity and ends at the distributed denial of service (DDoS) attack. The existing studies mostly focus on detecting botnet attacks after the IoT devices get compromised, and start performing the DDoS attack. Similarly, the performance of most of the existing machine learning based botnet detection models is limited to a specific dataset on which they are trained. As a consequence, these solutions do not perform well on other datasets due to the diversity of attack patterns. Therefore, in this work, we first produce a generic scanning and DDoS attack dataset by generating 33 types of scan and 60 types of DDoS attacks. In addition, we partially integrated the scan and DDoS attack samples from three publicly-available datasets for maximum attack coverage to better train the machine learning algorithms. Afterwards, we propose a two-fold machine learning approach to prevent and detect IoT botnet attacks. In the first fold, we trained a state-of-the-art deep learning model, i.e., ResNet-18 to detect the scanning activity in the premature attack stage to prevent IoT botnet attacks. While, in the second fold, we trained another ResNet-18 model for DDoS attack identification to detect IoT botnet attacks. Overall, the proposed two-fold approach manifests 98.89% accuracy, 99.01% precision, 98.74% recall, and 98.87% f1-score to prevent and detect IoT botnet attacks. To demonstrate the effectiveness of the proposed two-fold approach, we trained three other ResNet-18 models over three different datasets for detecting scan and DDoS attacks and compared their performance with the proposed two-fold approach. The experimental results prove that the proposed two-fold approach can efficiently prevent and detect botnet attacks as compared to other trained models.

INDEX TERMS Internet of Things, IoT botnet, botnet detection, IoT botnet attacks, IoT botnet DDoS attack, DDoS attack prevention, DDoS attack, IoT DDoS attack, botnet attack, botnet DDoS.

I. INTRODUCTION

Internet of Things (IoT) revolutionized the technology by enabling real-world objects/things to connect and communicate with each other over the internet to luxuriate human life [1], [2]. Over the past few years, the adoption of smart IoT devices like smart cameras, smart TV, smart wearables, smart toys, smart bulbs, etc., is exponentially increasing in our daily life [3], [4]. Therefore, this new emerging trend in the field of computing has empowered our everyday life objects to connect and communicate with each other without human intervention. Despite the IoT devices are helping us in a lot of areas, these devices have negligible or very limited security features [3]. Furthermore, many IoT devices come with a fixed key or hard-coded default username and password, which a user cannot change [5]. These security pitfalls make

The associate editor coordinating the review of this manuscript and approving it for publication was Zhibo Wang.

it easy for hackers to exploit these insecure IoT devices and get control over them [4].

The recent trends reveal that the cyber-attacks increasing day by day with the rapid increase of insecure IoT devices [6]. Among the recently reported cyber-attacks, botnet and distributed denial of service (DDoS) attacks are the most prevalent attacks, which are increased both in frequency and magnitude over the last decade [4], [6]. A botnet attack is a cyber-attack in which an attacker first scans a network to look for weakly secured or vulnerable (IoT) devices. After analysing the scanning information, the attacker targets vulnerable (IoT) devices to install a bot program into them through malware [7].

The installed bot program connects the infected devices with a central server or a peer network from where the further commands are sent to them to perform different malicious activities like sending spams, flooding DDoS [6], [8], etc., from plenty of infected IoT devices over the target server, website, etc. Once an IoT device gets infected and becomes part of a botnet, then the attacker uses the infected device to perform DDoS attacks.

The botnet attack is not only a serious threat to insecure IoT devices but also a crucial threat to the whole internet [6]. With the advent of the Mirai botnet attack in 2016, the IoT botnet attacks are continuously escalating [9]. After the public disclosure of the Mirai botnet source code, many variants and imitators of Mirai botnet have been evolved [9]. These new variants and imitators have infected millions of IoT devices [3], [9] and wreaked ever large and catastrophic DDoS attacks like GitHub [10], AWS [11], etc., over the past few years.

Nowadays, attackers can easily locate insecure IoT devices via online services such as Shodan [12], Censys [13], etc. These online search engine services provide a huge amount of information to attack insecure IoT devices [9]. By compromising the insecure IoT devices, an attacker can perform several cyber-attacks such as spamming, phishing, DDoS [6], [8], [9], etc., to wreak havoc against the other resources on the Internet. Some recent studies exposed that IoT devices are much prone to botnet and DDoS attacks, as a wide range of DDoS attacks are performed by compromised IoT devices [14], [15]. Likewise, Gartner recently predicted that 25% of the cyber-attacks are posed due to the insecure IoT devices [16].

In order to secure the insecure IoT devices to become a bot and perform different DDoS attacks, there must be an efficient security system to detect IoT bots. The existing botnet and DDoS attack detection techniques are divided into two categories, i.e., host-based techniques and network-based techniques [17]. Due to the resource constraint nature (i.e., limited memory, battery, and compute power) of IoT devices, the host-based solutions are not feasible for IoT devices [1], [17]. However, the network-based solution is a better way to protect the IoT devices and network from these devastating cyber-attacks. The network-based techniques are subdivided into three main types [18]–[22]:

1) **Signature-based detection method:** relies on matching the network traffic with some specific rules defined in the rule database to detect and prevent potential attacks.
2) **Anomaly-based detection method:** analyses the normal behaviour of network traffic and builds a baseline profile of each device communicating in the network Any significant deviation from the baseline is considered as an anomaly. The anomaly-based detection method is further classified into two subtypes:
   - **Statistics-based detection:** These methods detect anomalies based on a statistical distribution of intrusions.
   - **Machine learning-based detection method:** detects abnormalities based on packet and payload features. These methods mainly detect and prevent potential attacks using machine learning models.
   - **Knowledge-based detection method:** detects anomalies based on the profile or previous knowledge of a network. The profile or previous knowledge of the network is generated under different test cases to detect abnormalities in the network [22].
3) **Specification-based detection method:** performs intrusions detection based on the specifications or rules defined by a user [22].

The major drawback of the signature-based detection method is that it only detects the known threats for which the rules are available in its rules' database [20], [21]. On the other hand, the stateful protocol-based detection methods have limited ability to inspect the encrypted traffic. However, the traffic behaviour analysis, i.e., anomaly detection is very effective in both analysing the encrypted traffic and detecting the unknown attacks [19]. In case of anomaly detection methods, the machine learning approach has shown tremendous performance in recent years. The machine learning-based detection methods are trained on datasets to learn and distinguish the behaviour and pattern of normal and attack traffic [20], [21]. Henceforth, by learning the normal and attack traffic patterns, the machine learning models are useful to detect new botnet and DDoS attacks that are derived variants or imitators of the existing botnet and DDoS attacks. The existing botnet attack detection methods detect the botnet after the IoT devices are compromised by some malware and start performing malicious activities as directed by the botmaster. Moreover, the performance of most of the existing machine learning based botnet detection models is limited to a specific dataset on which they are trained [6]. This is due to the fact that different datasets contain different types of botnet attacks. Further, the features used for detecting botnet attacks from one certain dataset, are not adequate to efficiently detect the botnet attacks from other datasets due to the diversity of botnet attacks [6]. As a consequence, these solutions do not perform well on other datasets due to the diversity of attack patterns [6]. However, in order to protect the IoT devices from being compromised, there is a crucial need for providing a protection mechanism to safeguard the IoT devices from botnet and DDoS attacks during the premature stage (i.e., scanning) of the botnet attack. Therefore, in this work, we propose a novel two-fold approach to prevent a botnet attack during the premature stage (i.e., scanning attack) and to detect DDoS attack in IoT network in case an attacker compromises an IoT device and start performing a DDoS attack. As discussed earlier that an attacker can use the bot-infected IoT devices to perform different malicious activities like sending spam emails, flooding DDoS [6], [8], etc., however, in this work, we focus on detecting DDoS attacks performed by bot-infected IoT devices. The proposed two-fold approach uses a state-of-the-art deep learning model, i.e., ResNet which is first trained for detecting the scanning activity and then trained for detecting the DDoS attack performed by the attacker or compromised IoT devices towards or outside the network.

For preventing the IoT devices and network from IoT botnet attacks, in the first fold, we trained the ResNet-18 [23] model for scanning attack detection so that it can detect the premature attack stage and notify about the malicious attempt before an attacker goes to further steps for compromising the IoT devices. On the other hand, in the second fold, we trained the ResNet-18 [23] model for DDoS attack detection to detect and mitigate the botnet attack, in case an attacker invades the scanning attack detection model, install malware on IoT devices and starts performing DDoS attacks. The key contributions of this work are as follows:

- We analysed the frequently used scanning and DDoS attack techniques and produced a generic dataset by generating 33 types of scan and 60 types of DDoS attacks. In addition, we partially integrated the scan and DDoS attack samples from three publicly-available datasets for maximum attack coverage for better training of machine learning algorithms.
- We proposed a two-fold machine learning approach to prevent and detect both inbound and outbound botnet attacks in the IoT network environment. The proposed two-fold approach prevents IoT botnet attacks by detecting the scanning activity, while it detects the IoT botnet attack by identifying the DDoS attack.
- Finally, to demonstrate that the performance of the proposed two-fold approach is not limited to a single dataset, we trained three ResNet-18 [23] models over three different datasets and compared their performance with the proposed two-fold approach for detecting and preventing IoT botnet attacks.

The rest of the paper is organized as follows: Section II presents a review of some existing work for botnet attack detection. Section III describes some background knowledge needed to understand the botnet attacks basics. Section IV explains the proposed methodology to prevent and detect IoT botnet and DDoS attacks. Section V discusses the experimental setup and results of the proposed two-fold approach to prevent and detect IoT botnet attacks. Lastly, Section VI concludes the paper.

TABLE 1. List of terminologies and acronyms used in the text.

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

To date, several techniques have been proposed for botnet attack detection. The existing botnet detection techniques are broadly divided into two types, i.e., graph-based techniques and flow-based techniques [4]. In the graph-based botnet detection techniques, all the communication nodes of a network are analysed to detect the anomalies which communicate differently as compared to the neighbour nodes [24]. On the other hand, in the flow-based botnet detection approach, both the inbound and outbound traffic statistics, i.e., features are monitored by the machine learning algorithms which detects the botnet attacks based on the traffic pattern resemblance.

Nguyen *et al.* [16] proposed a graph-based approach to detect the IoT botnet via printing string information (PSI) graphs. The authors used PSI graphs to get high-level features from the function call graph and then trained a convolution neural network (CNN), a deep learning model, over the generated graphs for IoT botnet detection. Likewise, Wang *et al.* [24] proposed an automated model named as BotMark. Their proposed model detects botnet attacks based on a hybrid analysis of flow-based and graph-based network traffic behaviours. The flow-based detection is performed by k-means, which calculates the similarity and stability scores between flows. While the graph-based detection uses the least-square technique and local outlier factor (LOF) which measures anomaly scores. Similarly, Yassin *et al.* [25] proposed a novel method that compromises a series of approaches such as the utilization of the frequency process against registry information, graph visualization and rules generation. The authors investigated the Mirai attacks using the graph-theoretical approach. In order to identify similar and dissimilar Mirai patterns, the authors used directed graphs. The proposed approach only focuses on the Mirai attack.

Almutairi *et al.* [27] proposed a hybrid botnet detection technique that detects new botnets implemented on three levels, i.e., host level, network level and a combination of both. The authors focused on focused HTTP, P2P, IRC, and DNS botnet traffic. The proposed technique consists of three components: host analyser, network analyser, and detection report. The authors used two machine learning algorithms, i.e., Naïve Bayes and a decision tree for traffic classification. Similarly, Blaise *et al.* [28] proposed a bot detection technique named BotFP, for bot fingerprinting. The proposed BotFP framework has two variants, i.e., BotFP-Clus which groups similar traffic instances using clustering algorithms and BotFP-ML is designed to learn from the signatures and identify new bots using two supervised ML algorithms, i.e., SVM and MLP. Likewise, Soe *et al.* [30] developed a machine learning-based IoT botnet attack detection model. The proposed model consists of two stages: a model builder and an attack detector. In the model builder stage, data collection, data categorization, model training and feature selection are performed step by step. While in the attack detector stage, the packets are first decoded and then the features are extracted in the same way as in the model builder phase. Finally, the features are passed to the attack detector engine where artificial neural network (ANN), J48 decision tree, and Naïve Bayes machine learning models are used for botnet attack detection.

Sriram *et al.* [31] proposed a deep learning-based IoT botnet attack detection framework. The proposed solution specifically considered network traffic flows, which are further converted into feature records and then passed to the deep neural network (DNN) model for IoT botnet attack detection. Nugraha *et al.* [32] evaluated the performance of four deep learning models for botnet attack detection by performing a couple of experiments. The experimental results revealed that CNN-LSTM outperformed all deep learning models for botnet attacks detection.

Parra *et al.* [33] proposed a distributed deep learning framework based on cloud computing. Their framework is designed to detect phishing and IoT botnet attacks. Their model consists of two machine learning models: (1) a distributed CNN (DCNN) for detecting URL based attacks directed to a client's IoT devices, (2) a recurrent neural network (RNN) and an LSTM network model for detecting Botnet attacks at the backend. Pektacs and Acarman [34] performed botnet detection using deep learning on network flow traffic. The proposed deep neural network was deployed to classify the traffic as benign or malicious. For making the performance of the model better, hidden layers and neurons are investigated. The proposed model has achieved 99% accuracy. Likewise, Ahmed *et al.* [35] proposed a deep learning model for botnets attacks detection.

Maeda *et al.* [36] proposed the botnets attack detection via deep learning on software-defined network (SDN). For botnet detection, the authors trained the deep learning model using the data collected on flow-based traffic from the botnets and then evaluated the detection accuracy. The authors used a multi-layer perceptron (MLP), a deep learning model, to detect infected IoT devices. Similarly, Meidan *et al.* [18] developed a novel IoT botnet attack detection technique via deep auto-encoder. For the malicious network traffic, nine IoT devices were infected with well-known IoT botnets, Mirai and BASHLITE. The authors trained deep auto-encoders separately for each IoT device on both benign and attack traffic.

Bovenzi *et al.* [37] proposed a hybrid two-stage intrusion detection system (IDS) for the IoT environment. Their proposed approach first detects the anomalies from the network traffic, while in the second stage they classify the anomalies into attack classes. The authors used a multi-modal deep auto-encoder for anomalies detection, while used three machine learning classifiers to classify anomalies detected in the first stage. Likewise, Mirsky *et al.* [39] also used auto-encoders and proposed a plug and play network IDS, i.e., Kitsune to detect anomalies on local network traffic using an unsupervised learning approach. The authors used a self-generated botnet attack dataset and evaluated the performance in both online and offline modes. Their proposed solution achieved good performance comparable to offline anomaly detectors.

Table 2 summarizes the distinctive characteristics of the works discussed above. It can be observed that most of the existing botnet detection approaches used traffic flow-based machine learning approaches for botnet attacks detection. Moreover, most of these solutions do not perform well on other datasets due to the diversity of attack patterns [6]. However, these existing techniques only detect the botnet attack after the IoT devices get compromised and start performing malicious activities like DDoS attacks. In order to prevent the IoT devices from being compromised, in this study, we propose a two-fold approach that detects the attacker's malicious activities at the premature stage (i.e., scanning) of a botnet attack. Moreover, the proposed two-fold approach is

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

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

also capable of detecting DDoS attacks if a compromised IoT device starts performing malicious DDoS activities.

III. PRELIMINARIES

A. COMPONENTS OF AN IoT BOTNET

In general, a botnet comprises four components. These components include the bot program, zombie device, bot-master and command and control (C&C) server.

The botnet attack starts with the bot-master, which can be an attacker itself or an automated program written by the attacker. The bot-master scans the target IoT devices connected over the internet. Based on the scanning results, the bot-master exploits the vulnerable IoT devices and installs a bot program in vulnerable devices. The bot program establishes a connection with a bot-master or C&C server to receive the instructions for performing malicious activities. A brief description of each component is given in the following subsections:

1) BOT PROGRAM

A bot program is a malware installed on an infected device by an attacker. The bot program resides in the victim IoT device and establishes a connection with the C&C server or bot-master to receive the instructions for performing malicious activities like sending spams, performing flooding attacks [6], [8], etc.

2) ZOMBIE DEVICE

A physical victim device, on which a bot program is installed by the attacker, is called a zombie device. In the IoT scenario, these devices include smart cameras, smart TVs, smart wearables, etc.

3) BOT-MASTER

A bot-master or a bot header is the main controller of a botnet. It can be a hacker or an automated program controlled by the hacker to organize the botnet attacks. The bot-master works as the main operator of a botnet that issues commands to the C&C server (in client-server architecture) or specific bots (in peer-to-peer architecture).

4) C&C SERVER

The C&C server is the central computer that controls the zombie devices based on the control signals received from the bot-master. The C&C server is not a compulsory component of every botnet. In the case of a peer-to-peer network, the bot-master directly sends control signals to zombies.

B. LIFE-CYCLE OF AN IoT BOTNET

The botnet attack is a multi-stage attack [9]. A vulnerable IoT device passes through five stages to become a bot for performing malicious activities like sending spams, performing DDoS [9], etc. These stages are also referred as the botnet life cycle. These stages include scanning, malware injection, botnet connection, command execution, and maintenance & up-gradation as shown in Fig. 1. Scanning is the initial stage of a botnet life cycle in which the attacker collects information to proceed with further steps. The information collected by scanning helps an attacker to exploit the vulnerability which allows him/her to inject malware into the target device.

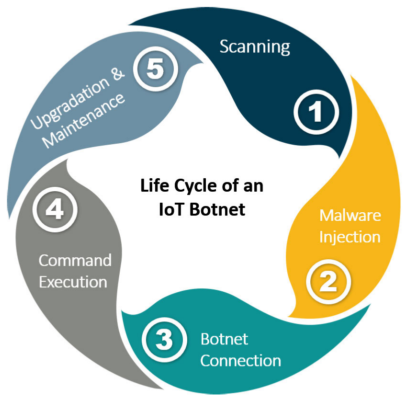

FIGURE 1. Five stages of an IoT botnet attack life cycle.

Afterwards, the injected malware establishes the connection with the bot-master and executes the instructions received from the bot-master. Finally, the bot-master maintains and upgrades the infected devices in order to perpetuate the infection for future use. All these stages are defined in the following subsections:

1) SCANNING
Scanning is the preliminary step of a botnet life cycle. In this stage, the attacker/bot-master scans the target network or target device to collect the initial information about the services, protocols, OS, etc. of the target device. The attackers use different techniques for scanning an IoT network or target device. Some popular tools used for scanning include: NMap [40], ZMap [41], Masscan [42], OpenVAS [43], etc.

2) MALWARE INJECTION
Once the attacker collects information about the target network/device, next he/she applies different exploitation methods to find the vulnerabilities of the target device/network. The successful exploitation, allows an attacker to inject the malware into a target device. Besides the vulnerabilities' exploitation, the attacker can trap the victim by sending malware through phishing, email attachments, etc. The victim unknowingly downloads the malicious software from the phishing website or email attachments, which helps the attacker to proceed with further steps. At first, the attacker installs a shellcode on the victim device. This step is also called an initial infection. The running shellcode fetches some more details about the victim device and sends it to a central server from where a bot program binary is downloaded along with some additional configurations. Afterwards, the bot program is installed with respect to the target device properties [44], [45]. This step is also called a secondary injection. When the bot program is installed in the victim device, it becomes a 'zombie' [45].

3) BOTNET CONNECTION
When the bot program starts running, it establishes a communication channel with the bot-master or C&C server to deem a valid bot. The initial connection attempts done by the zombie with the C&C server to receive further commands from the bot-master are called as rally [45]. When the malware is injected, it starts executing as per the attacker's strategy. The attacker can code the malware to run as a Trojan and run the malware based on some event. The running malware connects the target machine with the bot-master, from where it receives the commands to do further steps.

4) COMMAND EXECUTION
Once an infected device gets connected with a bot-master or C&C server, it becomes part of a botnet army. Afterwards, it waits for the C&C server's commands to perform malicious activities as instructed by the bot-master. This phase is also called the waiting phase. The malicious activities include scanning for new bots, sending spams, performing DoS attacks [6], [8], [46], etc. When many infected devices are connected with the bot-master, the bot-master sends commands to these infected devices to do flooding/DDoS attacks on a target server/network.

5) UPGRADATION & MAINTENANCE
A bot program installed in a victim machine needs to be updated and maintained with time in order to remain undetected in the victim machine. This step has great importance for an attacker to perpetuate the malware infection so that the infected machine can be used in future attacks/malicious activities as well.

The life cycle of a traditional and IoT botnet is similar [5]. The difference is only in target devices. In traditional botnet attacks, the target of an attacker is to victimize the computers, servers, etc. while in the case of IoT botnet attacks, the target of an attacker is to victimize the IoT devices like smart cameras, smart TVs [5], etc.

IV. PROPOSED METHODOLOGY
In this work, we proposed a novel two-fold machine learning approach to prevent and detect botnet attacks in IoT networks. In the first fold, we trained a state-of-the-art deep learning model, i.e., ResNet-18 [23] for detecting the (pre-attack stage) scanning activity to protect the IoT network from botnet attacks. While in the second fold, we trained another ResNet-18 [23] model for detecting the DDoS attack that attackers perform after compromising the weakly-secured IoT devices.

As discussed earlier that the lifecycle of a botnet attack consists of five stages, i.e., scanning, malware injection, botnet connection, command execution, and maintenance & up-gradation. Scanning is the initial premature attack stage of the botnet attack. The proposed methodology stops an attacker during the scanning activity so that an attacker cannot proceed to further attack stages. Thus, the proposed methodology prevents botnet attacks by detecting the scanning attack activity while it detects the botnet attack by identifying the DDoS attack for both inbound and outbound traffic.

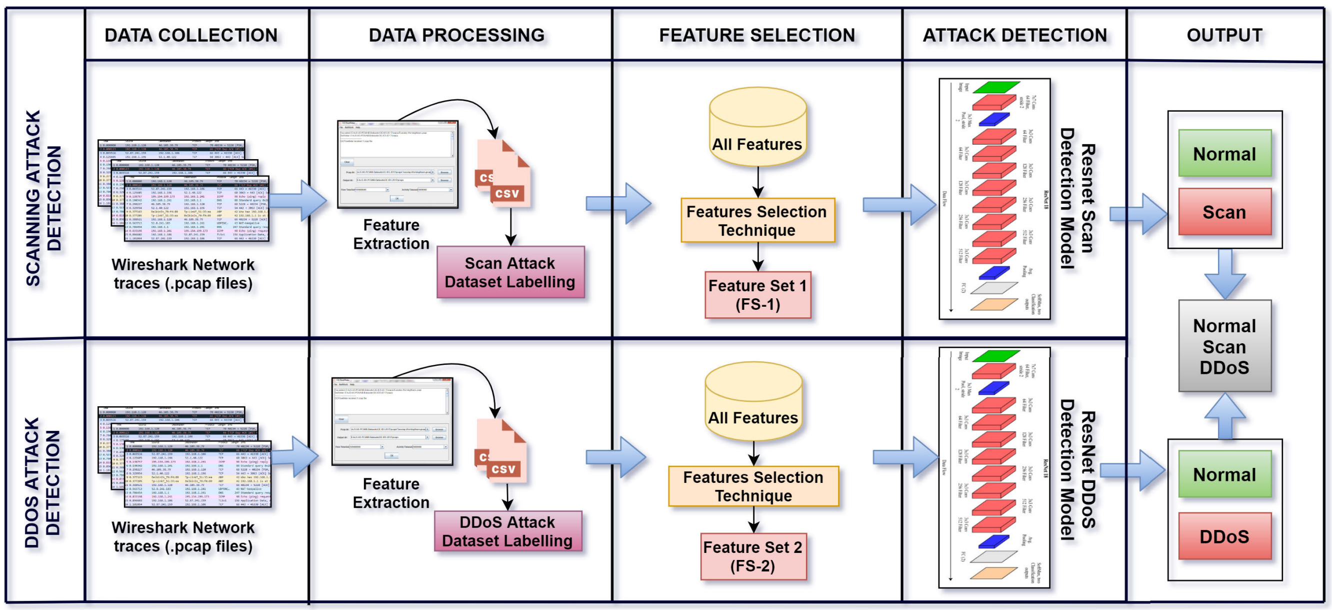
DATA COLLECTION
DATA PROCESSING
FEATURE SELECTION
ATTACK DETECTION
OUTPUT
SCANNING ATTACK DETECTION
DDOS ATTACK DETECTION
Wireshark Network traces (.pcap files)
Feature Extraction
csv
Scan Attack Dataset Labelling
DDoS Attack Dataset Labelling
All Features
Features Selection Technique
Feature Set 1 (FS-1)
Feature Set 2 (FS-2)
Resnet Scan Detection Model
ResNet DDoS Detection Model
Normal
Scan
DDoS
Normal Scan DDoS

FIGURE 2. The proposed Two-fold approach to prevent and detect IoT botnet attacks.

Each fold of the proposed approach passes through five major stages for scanning and DDoS attacks detection, as illustrated in Fig. 2. In the first stage, we generated and captured the scanning and DDoS attack traffic (in.pcap format) in order to use it for training the machine learning models. In the second stage, we converted these network packet traces (.pcap files) into flows, then extracted the features and stored them in.csv files. Further, we labelled the dataset with labels such as 'normal' for benign traffic, 'scan' for scanning traffic and 'DDoS' for the DDoS attack traffic. In the third phase, we applied the Logistic Regression (LR) feature selection technique to optimize the performance of the machine learning model using minimum unique features. We used the LR feature selection technique due to its efficient performance in the existing literature [6], [8], [17]. Moreover, it is fast, simple, and has low complexity as compared to other feature selection techniques [6], [8], [17]. In the fourth stage, we trained two ResNet-18 [23] models over the resultant feature vector, once for scanning and then for DDoS attack detection. Finally, we test the performance of the trained machine learning models in order to validate their performance for real-time attack scenarios.

As discussed earlier that an IoT botnet attack initiates with the scanning activity and ends at the DDoS attack. Therefore, the proposed framework consists of two machine learning models, i.e., one for preventing the IoT botnet attacks while the other for detecting the DDoS attacks. For preventing the IoT devices and network from IoT botnet attacks, we trained the ResNet-18 [23] model on scanning attacks and normal traffic dataset so that it can prevent the IoT botnet attacks during the premature attack stage, i.e., scanning stage. Since this model prevents the IoT botnet attacks by detecting the scanning attacks performed by the attacker in the premature attack stage to collect information about the vulnerable IoT devices, therefore, we called this model as ResNetScan-1 model. On the other hand, in case if an attacker invades the scanning attack detection model, installs malware on IoT devices and starts performing DDoS attacks. Then for detecting the DDoS attack performed through the compromised IoT devices, we trained another ResNet-18 [23] model over the DDoS attack and normal traffic dataset so that it can detect the DDoS attack activity. We called this model a ResNetDDoS-1 model.

Finally, we integrated both the ResNetScan-1 and ResNetDDoS-1 models to classify the incoming network data as scan, DDoS, or normal as shown in Fig. 2. The following subsections describe all the steps followed to train and test the proposed machine learning models for preventing and detecting IoT botnet attacks.

A. SCANNING ATTACK DETECTION

In the first fold, we trained a ResNet-18 [23] model for scanning attack detection by following the five steps as mentioned previously. These steps are described in the following sections.

1) DATA COLLECTION

The data collection is the preliminary step of the proposed methodology for scanning attack detection. In this step, we first analysed some existing techniques and approaches [47]–[51] that the attackers commonly use for scanning the IoT network and devices to collect the information.

Based on the literature review [47]–[51], we selected eleven different scanning methods that attackers widely use for gathering information about the vulnerable IoT during the premature attack stage. These scanning techniques include SYN scan, FIN scan, ACK scan, NULL scan, SYN-ACK scan, FIN-ACK scan, XMAS scan, UDP scan, TCP Window scan, TCP connect scan, and Banner grabbing and are performed using three widely used scanning tools, i.e., Nmap [40], Hping3 [52], and Dmitry [53]. In order to generate the scan traffic for this work, we performed these scanning attacks on two of the lab servers using three different scanning approaches which include horizontal scan, vertical scan, and box scan. So, a total $(11 \times 3) = 33$ types of scanning attacks were performed to generate and collect scanning traffic. We performed scanning attacks by installing three widely used scanning tools, (i.e., Nmap [40], Hping3 [52], and Dmitry [53]), on an Ubuntu machine with a Core i7 processor and 8 GB RAM. Afterwards, we write some python scripts to execute different scanning commands. While performing the scanning attacks, we captured the network packets into.pcap files using the Wireshark tool [54]. We call this self-generated dataset as ScanLab dataset.

Besides generating and collecting the network packets from our experimental setup, we also obtained scan and normal traffic samples from three publicly-available datasets which include CICIDS-19 [55], CICIDS-17 [56] and Bot-IoT [38] dataset. The scan and normal samples of these three publicly-available datasets are acquired to compare the performance of the ResNet-18 [23] model trained over the ScanLab dataset, with ResNet-18 [23] models trained over other datasets for scanning attack detection.

2) DATA PRE-PROCESSING

After capturing the scanning traffic, we need to pre-process the data. In the pre-processing step, we first extracted the features from the captured.pcap files of the ScanLab dataset using the CICFlowmeter [57] Tool. The CICFlowmeter [57] tool reads a given.pcap file and extracts more than 60 flow features for each flow which is identified based on five-tuple which include source IP, destination IP, source port, destination port, and protocol. The details of these features extracted by CICFlowmeter [57] are given at [58]. The CICFlowmeter [57] results a.csv file which consists of the flow features of a given.pcap file. The resultant data is unlabelled. So based on the IP addresses used for scanning, we labelled them as scan while the rest of the network traffic is labelled as normal traffic.

Similarly, we pre-processed CICIDS-19 [55], CICIDS-17 [56] and Bot-IoT [38] dataset and labelled the resultant.csv files with respect to the description of these datasets. Eventually, we partially integrated the scan attack samples (by randomly selecting the 50K samples from these three datasets) with the ScanLab dataset for maximum attack coverage for better training of machine learning algorithms.

3) FEATURES SELECTION

Once we extracted the features from all.pcap files, the next step is to select the useful features that can better help a machine learning model for distinguishing the normal and scan traffic. For features selection, we used the LR algorithm due to its better performance in existing research studies [6], [8]. So, by using the LR algorithm, we first selected the top 20 features from each dataset including the ScanLab dataset and three other selected datasets. Afterwards, we performed a frequency analysis—as done in [6]—of the features selected by the LR algorithm from the ScanLab dataset and all three selected datasets.

Based on the frequency analysis of the features selected by the LR algorithm, we found 15 most frequently selected features and named them as features set 1 (FS-1) as displayed in Figure 2 and enlisted in Table 3. The 15 features enlisted in Table 3 are selected from each dataset in order to use them in the subsequent stages of scanning attack detection.

4) TRAINING ML MODEL FOR SCAN DETECTION

After selecting the useful features, we split each dataset into the train, validation, and test set. For this purpose, we randomly selected 60% data for training, 20% data for validation and 20% data for testing to avoid overfitting and for efficiently training the ML model. Both the training set and validation set are used during the training phase. The training set is used to train the machine learning model. In order to efficiently train and better optimize the weights of an ML model, we validate the trained model on the validation set after each epoch based on which the optimizer algorithm updates the weights of the ML model. Finally, when the ML model completes its training, we test its performance over unseen data, i.e., test set. As mentioned earlier that we used the ResNet-18 [23] model and first trained it over the train set of the ScanLab dataset.

Originally, the ResNet-18 [23] model is designed to classify the image processing and computer vision problems [4] which consist of images, i.e., high dimensional arrays. So, before starting the training, we need to convert the data into high dimensional arrays since the ResNet-18 [23] model is prone to overfit at low dimension data [4], [59]. Therefore, we first converted the whole ScanLab dataset into high dimension arrays of size $15 \times 15 \times 1$ and saved them as images by following the method described in [4]. Similarly, we converted the other three datasets into greyscale images of size $15 \times 15 \times 1$. Furthermore, we also need to mention the hyperparameters, i.e., learning rate, batch size, number of epochs, and optimizer. The learning rate controls the updates of ML model weights based on the estimated error after each epoch. The epochs tell about the number of iterations for which an ML model is trained. The batch size divides the given dataset into small chunks for fast and better training. While the optimizer set the better attributes of weights of a neural network to improve the speed and performance of a machine learning model. So, we set the learning rate

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

as 0.01 with batch size 100 with 12 epochs, and selected stochastic gradient descent (SGD) optimizer.

After setting the hyperparameters, we passed the training and validation set obtained from the ScanLab dataset to the ResNet-18 [23] model, and started the training. After the ResNet-18 [23] model completed the training, we saved the trained model as ResNetScan-1. The similar method was followed to train the ResNet-18 [23] model over the other three datasets. We saved the resultant models as ResNetScan-2, ResNetScan-3, ResNetScan-4 model obtained after training the ResNet-18 [23] model on CICIDS-19 [55], CICIDS-17 [56], and Bot-IoT [38] dataset respectively.

5) TESTING AND AFFIRMATION

Once the trained models are saved, we then test the performance of the trained ResNet model over the test set. The test set consists of samples that are separated before training the ML model. As the test set is unknown for the trained model, so in order to check the performance of the trained model over the test set we evaluated the performance of the trained model over four commonly used performance metrics. These performance metrics are described in Section V. The results of the trained model over the test set for scanning attack detection are also given Section V.

In order to affirm the effectiveness of the proposed methodology, we test the performance of all scan detection models in four phases. In the first phase, we test the proposed ResNetScan-1 model over the three datasets that were not used in its training, i.e., CICIDS-19 [55], CICIDS-17 [56], and Bot-IoT [38] dataset. Similarly, in the second phase, we test the ResNetScan-2 model over the three datasets that were not used in its training, i.e., ScanLab, CICIDS-19 [55], and Bot-IoT [38] dataset. Likewise, in the remaining phases, the other two ResNetScan models, i.e., ResNetScan-3 and ResNetScan-4 are tested over datasets that were not used in their training. Finally, we compared the performance of all ResNetScan models on each dataset.

B. DDoS ATTACK DETECTION

In the second fold of the proposed approach, we trained a ResNet-18 [23] model for DDoS attack detection. As mentioned earlier, the DDoS attack detection model is proposed to detect the DDoS attacks in case if an attacker invades the scanning attack detection model, installs malware on IoT devices and starts performing DDoS attacks.

In order to develop the DDoS detection model, we followed the five steps as mentioned earlier. These steps are described in the following sections.

1) DATA COLLECTION

Like the scanning attacks, in this step, we first analyzed the existing DDoS attack techniques that are commonly used by the attackers to perform DDoS attacks. There exist a vast literature on DDoS attacks as compared to the scan attacks, therefore, we reviewed some recent studies [60]–[62] on DDoS attacks to analyze the DDoS attack types.

Based on the analysis of different DDoS attack techniques, we performed 60 different types of DDoS attacks which include all 57 TCP flag based attacks given in Table 4, UDP, ICMP, and HTTP flooding attacks. All these DDoS attacks are performed using the Hping3 tool [52]. We write python scripts that execute different commands of Hping3 [52] to perform DDoS attacks on two of our Lab servers. While performing the DDoS attacks, we captured the network packets into.pcap files using the Wireshark tool [54]. We named this self-generated dataset as the DDoSLab dataset.

Besides generating and collecting the network packets from our experimental setup, we also obtained DDoS and normal traffic samples from three publicly-available datasets which include CICIDS-19 [55], CICIDS-17 [56] and Bot-IoT [38] dataset. These datasets are considered to compare the performance of the ResNet-18 [23] model trained over self-generated DDoSLab, with the ResNet-18 [23] model trained over other datasets for scanning attack detection.

2) DATA PRE-PROCESSING

After capturing the DDoS traffic into.pcap files, we extracted the features from the captured.pcap files of the DDoSLab dataset using the CICFlowmeter [57] Tool. Since the CICFlowmeter [57] results in an unlabelled.csv file which consists of the flow features of a given.pcap file. So based on the IP addresses used for the DDoS attack, we labelled them as DDoS while the rest of the network traffic is labelled as normal traffic.

Likewise, we pre-processed DDoS and normal traffic of CICIDS-19 [55], CICIDS-17 [56] and Bot-IoT [38] dataset and labelled the resultant.csv files with respect to the

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)				

description of these datasets. Further, we partially integrated the DDoS attack samples (by randomly selecting the 50K samples from these three datasets) with the DDoSLab dataset for maximum attack coverage for better training of machine learning algorithms.

3) FEATURES SELECTION

Once the features are extracted and the dataset is labelled, we then applied the LR algorithm on the labelled dataset in order to select the useful features that can better help a machine learning model for distinguishing the normal and DDoS traffic. So, using the LR algorithm, we first selected the top 20 features including the DDoSLab dataset and three other selected datasets. Afterwards, we performed a frequency analysis—as done in [6]—of the features selected by the LR algorithm from DDoSLab and all three selected datasets. Finally, based on the frequency analysis of the features, we found 15 most frequently selected features and named them as features set 2 (FS-2) as displayed in Figure 2 and enlisted in Table 3 that are selected from DDoSLab and the other three datasets.

4) TRAINING ML MODEL FOR DDoS DETECTION

After selecting the useful features, we then divided the dataset into train, validation, and test set. We randomly selected 60% data for training, 20% data for validation and 20% data for testing to avoid overfitting and for efficiently training the ML model. In order to train the ML model over the DDoS dataset, we selected the ResNet-18 [23] model due to its better performance in existing research works.

Before starting the training, we converted the whole DDoSLab dataset into high dimension arrays of size $15 \times 15 \times 1$ and saved them as images by following the method described in [4]. Similarly, we converted the other three datasets into greyscale images of size $15 \times 15 \times 1$. Furthermore, we set the values of hyperparameters, i.e., learning rate as 0.01 with batch size 64, 12 epochs, and selected SGD optimizer.

After setting the hyperparameters, we first passed the training and validation set extracted from the DDoSLab dataset to the ResNet-18 [23] model and started the training. Once the training is completed, we saved this trained model as ResNetDDoS-1. Similarly, we trained the three other ResNet-18 [23] models for other datasets. We saved the resultant models as ResNetDDoS-2, ResNetDDoS-3, ResNetDDoS-4 model obtained after training the ResNet-18 [23] model on CICIDS-19 [55], CICIDS-17 [56], and Bot-IoT [38] dataset respectively.

5) TESTING AND AFFIRMATION

Once the training of the ResNet-18 [23] model is complete, we then test the performance of the trained model over the test set. Since the test set is unknown for the trained model, so in order to check the performance of the trained model over the test set, we evaluated the performance of the trained model over four commonly used performance metrics. These performance metrics are described in Section V. The results of the trained model over the test set for DDoS attack detection are also given in Section V.

In order to authenticate the efficiency of the proposed methodology, we evaluated the predictions of all the trained models in four phases. In the first phase, we test the proposed ResNetDDoS-1 model over the three datasets that were not used in its training, i.e., CICIDS-19 [55], CICIDS-17 [56], and Bot-IoT [38] dataset. Likewise, we cross-validated the performance of other saved models over the dataset that were not during their training. Finally, we compared the performance of all ResNetDDoS models on each dataset.

V. RESULTS AND DISCUSSION

A. EXPERIMENTAL SETUP

As mentioned earlier that in this study, we first analysed the existing scanning and DDoS attacks techniques. Based on the analysis, we generated 33 types of scanning attacks traffic and 60 types of DDoS attacks traffic using 3 different network traffic generator tools, i.e., Nmap [40], Hping3 [52] and Dmitry [53]. All these tools were installed in a Core i7 machine with 8 GB RAM and having Ubuntu-18 operating system installed on it.

The generated network traffic was captured using the Wireshark tool [54] in.pcap format. After that, we extracted features from these.pcap files and performed labelling according to the IP addresses of the machines used in the experiment. Afterwards, we applied the feature selection techniques and split the dataset into train and test set to proceed with the proposed methodology with further steps as described 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 $\begin{bmatrix} 3\times3, & 64 \\ 3\times3, & 64 \end{bmatrix} \times 2$
conv3	28 x 28	$\begin{bmatrix} 3\times3, & 128 \\ 3\times3, & 128 \end{bmatrix} \times 2$
conv4	14 x 14	$\begin{bmatrix} 3\times3, & 256 \\ 3\times3, & 256 \end{bmatrix} \times 2$
conv5	7 x 7	$\begin{bmatrix} 3\times3, & 512 \\ 3\times3, & 512 \end{bmatrix} \times 2$
prediction	1 x 1	average pool, fc, softmax

Section IV. The proposed two-fold approach detects two cyber-attacks, i.e., scan attack and DDoS attack. Both of these attacks have different nature. Similarly, the best features resulted from the feature selection are different for both attacks, as enlisted in Table 3. Therefore, we trained two ResNet-18 [23] models separately for the better prevention and detection of IoT botnet attacks.

The ResNet-18 [23] is basically a convolutional network with eighteen layers which consists of 10 convolution layers and 8 pooling layers. Table 5 represents the architecture of the ResNet-18 [23] model used in this work. In total, our ResNet-18 [23] model had 11,185,666 computational parameters out of which 11,176,066 were trainable parameters while 9600 were non-trainable parameters. In order to train and test ResNet-18 [23] model for this work, we used Python3 and TensorFlow v2.2 library run over Google Colab environment. Table 6 enlists the execution time spent while training and testing the two separate ResNet-18 [23] models in the proposed two-fold approach. Based on the testing stats shown in Table 6, it can be verified that the ResNetDDoS-1 model takes (49.45/63668 =) 776.685 μs for testing an image while the ResNetScan-1 model takes (5.75/12546 =) 458.313 μs for testing an image. So, using a second model induced 1.59 times more delay (while considering the ResNetDDoS-1 as a single primary model) as compared to a single detection model.

B. PERFORMANCE EVALUATION

The proposed two-fold approach for preventing and detecting IoT Botnet attacks is evaluated based on the four commonly used performance parameters. These parameters include precision, recall, accuracy, and F1-score. These parameters are defined below. In order to calculate these performance parameters, we first define the following terms:

- **True Positive (TP):** The ML model truly predicted the attack flow as an attack.

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 ML model truly predicted the normal flow as normal.
- **False Positive (FP):** The ML model wrongly predicted the normal flow as an attack.
- **False Negative (FN):** The ML model wrongly predicted the is attack flow as normal.

1) **Accuracy:** It is defined as the ratio of correctly classified the attack flows as 'attack flow' (i.e., TP) and normal traffic flows as 'normal flow' (i.e., TN). Mathematically, it is defined as (1):

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \times 100 \tag{1}$$

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

$$Precision = \frac{TP}{TP + FP} \times 100 \tag{2}$$

3) **Recall** - It defines 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):

$$Recall = \frac{TP}{TP + FN} \times 100 \tag{3}$$

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

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

1) TEST SCENARIO 1: WHEN EACH ResNetScan MODEL IS TRAINED AND TESTED OVER SIMILAR DATASET

Table 7 summarizes the testing results of each ResNetScan model when tested over the test-set of the similar dataset on which it was trained for detecting the scanning attack traffic. The first row of the Table 7 shows the performance of the ResNetScan-1 model, which is obtained by training the ResNet-18 [23] model over the train-set extracted from the ScanLab dataset. When the ResNetScan-1 model is tested over a similar dataset (i.e., ScanLab dataset) on which it was trained, it resulted in 99.20% accuracy, 99.39% precision, 99.05% recall, and 99.22% f1-score for classifying the normal and scanning attack traffic.

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

Similarly, the ResNetScan-2 model is obtained after training the ResNet-18 [23] model over the CICIDS-19 [55] dataset. When the ResNetScan-2 is tested over the test-set of its respective dataset, i.e., CICIDS-19 [55] dataset, it showed 99.91% accuracy, 100% precision, 99.83% recall, and 99.92% f1-score for detecting the normal and scanning attack traffic. Likewise, the performance of other ResNetScan models for detecting the normal and scanning traffic over their respective datasets, is shown in Table 7. It can be observed that all the ResNetScan models performed well with accuracy, precision, recall, f1-score more than 98% when trained and tested for scanning attack detection. Overall, the ResNetScan-2 model outperformed all other models for correctly classifying the normal and scan traffic, when we evaluated its performance over the test-set of the similar dataset on which it was trained.

2) TEST SCENARIO 2: WHEN EACH ResNetScan MODEL IS TESTED OVER OTHER DATASETS

In this test scenario, we performed experiments to cross-validate the performance of each ResNetScan model over the test-set of other datasets. Table 8 displays the results of all experiments in which we test each ResNetScan model for normal and scan traffic detection. In the first experiment, we tested the ResNetScan-1 model (obtained after training the ResNet-18 [23] model over ScanLab dataset) over the test-set of CICIDS-19 [55], CICIDS-17 [56] and Bot-IoT [38] datasets. The experimental results manifest that the ResNetScan-1 model performed predominantly well over all three datasets for correctly classifying the normal and scan traffic. On average, the ResNetScan-1 model demonstrated 99.39% accuracy, 99.34% precision, 99.44% recall, and 99.39% f1-score which is equivalent to its performance when trained and tested over a similar dataset.

Similarly, in the second experiment, we tested the performance of the ResNetScan-2 model over the test-set of ScanLab, CICIDS-17 [56] and Bot-IoT [38] dataset. The experimental results reveal that on average, the ResNetScan-2 model resulted in 54.41% accuracy, 47.36% precision, 35.01% recall, and 40.23% f1-score as shown in Table 8. As a whole, the average performance of the ResNetScan-2 model over the other scan datasets decreased by 45.50%, 52.64%,

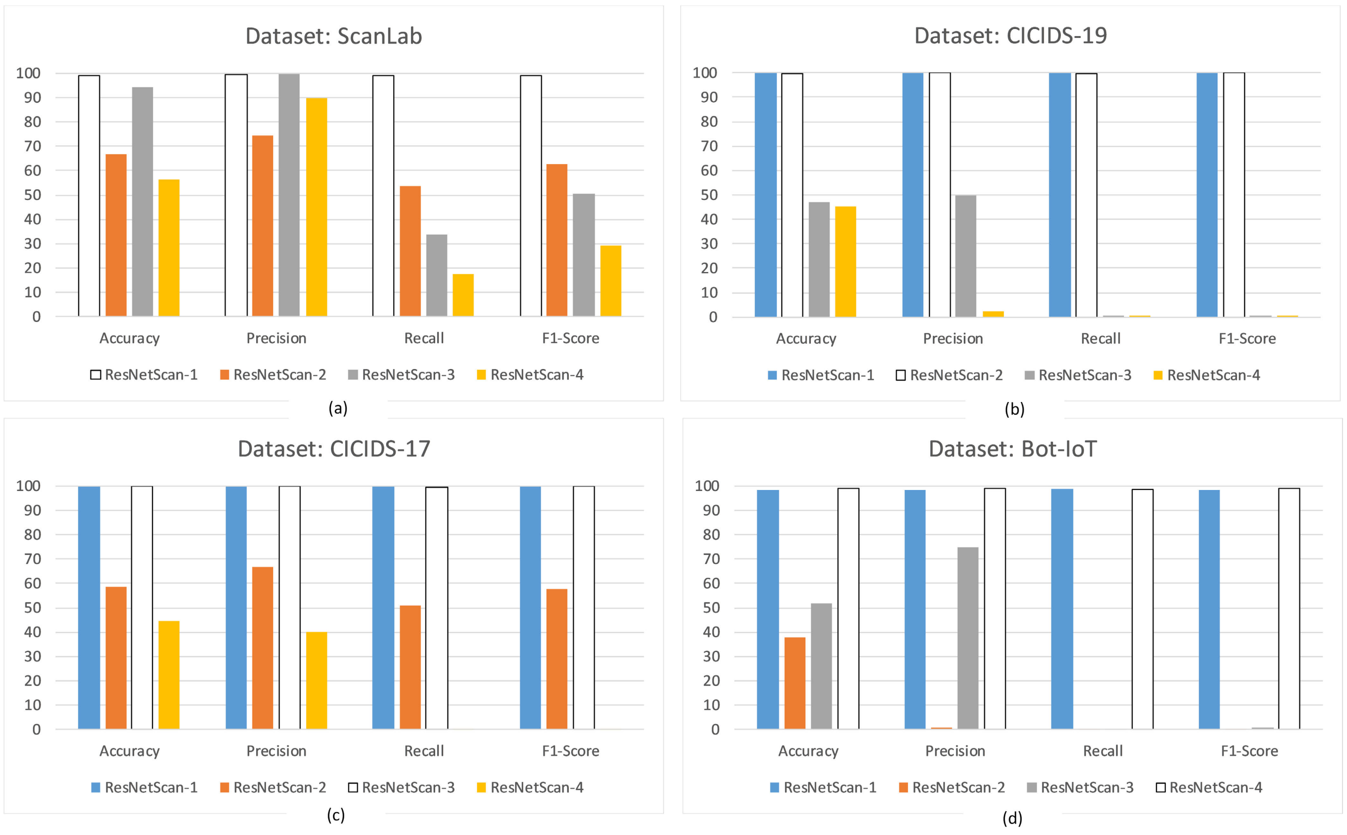

FIGURE 3. **Performance comparison of ResNetScan models on individual dataset. (a) Presents the results with the ScanLab dataset. (b) Presents the results with the CICIDS-19 dataset. (c) Presents the results with the CICIDS-17. (d) Presents the results with Bot-IoT dataset.**

64.82%, and 59.69% in case of accuracy, precision, recall, and f1-score respectively. In a nutshell, the performance of the ResNetScan-2 model was significantly downgraded as compared to its performance when trained and tested over a similar dataset.

In like manner, in the third experiment, we tested the performance of the ResNetScan-3 model over the test-set of ScanLab, CICIDS-19 [55] and Bot-IoT [38] dataset. The experimental results illustrate that on average, the ResNetScan-3 model demonstrated 64.30% accuracy, 74.94% precision, 11.33% recall, and 17.02% f1-score. These results indicate that the ResNetScan-3 aptly detected the normal traffic samples, however, it did not fairly detect the scan samples due to which the average f1-score drastically reduced as compared to average precision.

Eventually, in the fourth experiment, we tested the performance of the ResNetScan-4 model over the test-set of ScanLab, CICIDS-17 [56] and CICIDS-19 [55] dataset. The experimental results summarized in Table 8 present that the ResNetScan-4 model exhibited 48.75% accuracy, 44.12% precision, 5.88% recall, and 9.87% f1-score. These results also present that the ResNetScan-4 model insufficiently detected the scan attack samples, owing to which the average f1-score terrifically decreased as compared to average precision.

Fig. 3 (a)-(d) compares performance of all ResNetScan models with respect to each individual dataset. The white bars with black boundary line in Fig. 3 (a)-(d), presents the performance of the ResNetScan model which is trained and tested over a similar dataset while the other bars present the performance of other ResNetScan models on each dataset. From Fig. 3 (a)-(d), it can be noticed that the highest performance is achieved when a ResNetScan model is tested over the test-set of a similar dataset on which it was trained. However, if we further observe the Fig. 3 (a)-(d), it can be the proposed ResNetScan model, i.e., the ResNetScan-1 model accomplished the second-highest performance scores and outperformed all other ResNetScan models with remarkable accuracy, precision, recall, and f1-score. Thus, the experimental results prove that the proposed ResNetScan-1 model outperformed all other ResNetScan models when each ResNetScan model is tested over the dataset on which it was not trained. In case, an attacker invades the scan detection stage, compromises an IoT device, and starts performing the DDoS attack, then the attacker can be detected in the second stage. Further, if an IoT device or network becomes the victim of the DDoS attack (i.e., the device is under attack from outside the network) then the DDoS attack detection model will detect the inbound DDoS attack to stop the attacker from performing further damaging the network.

TABLE 9. Testing results when the ResNet model is trained and test over the similar dataset for DDoS attack detection.

Model Name	Dataset	Accuracy	Precision	Recall	F1-Score
ResNetDDoS-1	DDoSLab	98.57	98.63	98.42	98.52
ResNetDDoS-2	CICIDS-19 [55]	99.49	99.40	99.63	99.52
ResNetDDoS-3	CICIDS-17 [56]	97.49	95.59	92.83	94.19
ResNetDDoS-4	Bot-IoT [38]	99.70	99.58	99.6	99.59

TABLE 10. Testing results when the ResNetDDoS-1 model is test over other datasets for DDoS attack detection.

Testing Model	Testing Dataset	Accuracy	Precision	Recall	F1-Score
ResNetDDoS-1	CICIDS-19 [55]	98.99	98.91	99.18	99.04
	CICIDS-17 [56]	97.95	95.66	94.94	95.30
	Bot-IoT [38]	99.16	98.02	99.75	98.88
	Average Results	98.70	97.53	97.96	97.74
ResNetDDoS-2	DDoSLab	70.22	96.67	39.92	56.51
	CICIDS-17 [56]	78.41	63.01	3.30	6.27
	Bot-IoT [38]	68.58	93.56	16.19	27.60
	Average Results	72.40	84.41	19.80	30.13
ResNetDDoS-3	DDoSLab	62.56	94.74	24.07	38.39
	CICIDS-19 [55]	49.60	81.28	5.27	9.91
	Bot-IoT [38]	64.15	70.81	5.26	9.79
	Average Results	58.77	82.28	11.53	19.36
ResNetDDoS-4	DDoSLab	71.75	99.30	42.00	59.03
	CICIDS-19 [55]	47.53	70.31	0.23	0.44
	CICIDS-17 [56]	78.14	56.90	0.65	1.29
	Average Results	65.81	75.50	14.29	20.25

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

In this scenario, we performed experiments to compare the performance of each ResNetDDoS model when each ResNet-DDoS model is tested over the similar dataset on which it was trained. Table 9 presents the performance of all ResNetDDoS models for detecting the normal and DDoS attack traffic when each ResNetDDoS model is tested over the similar dataset on which it was trained. The first row of Table 9 displays the performance of the ResNetDDoS-1 model, which is obtained by training the ResNet-18 [23] model over the train-set extracted from the DDoSLab dataset. When the ResNetDDoS-1 model is tested over a similar dataset (i.e., DDoSLab dataset) on which it was trained, it resulted in 98.57% accuracy, 98.63% precision, 98.42% recall, and 98.52% f1-score for classifying the normal and DDoS attack traffic.

Similarly, the ResNetDDoS-2 model is obtained after training the ResNet-18 [23] model over the CICIDS-19 [55] dataset. The ResNetDDoS-2 showed 99.49% accuracy, 99.40% precision, 99.63% recall, and 99.52% f1-score for detecting the normal and scanning attack traffic when it is tested over the test-set of its respective dataset, i.e., CICIDS-19 [55] dataset. Likewise, the performance of other ResNetDDoS models for detecting the normal and DDoS traffic over their respective datasets, is shown in Table 9. It can be noticed that all the ResNetDDoS models performed well with accuracy, precision, recall, f1-score. Overall, the ResNetDDoS-2 model outperformed all other models for correctly classifying the normal and DDoS traffic, when we evaluated its performance over the test-set of the similar dataset on which it was trained.

4) TEST SCENARIO 4: WHEN EACH ResNetDDoS MODEL IS TESTED OVER OTHER DATASETS

In this test scenario, we performed experiments to cross-validate the performance of each ResNetDDoS model over the test-set of other datasets. Table 10 summarizes the results of all experiments in which we test each ResNet-DDoS model for normal and DDoS attack traffic detection. In the first experiment, we tested the ResNetDDoS-1

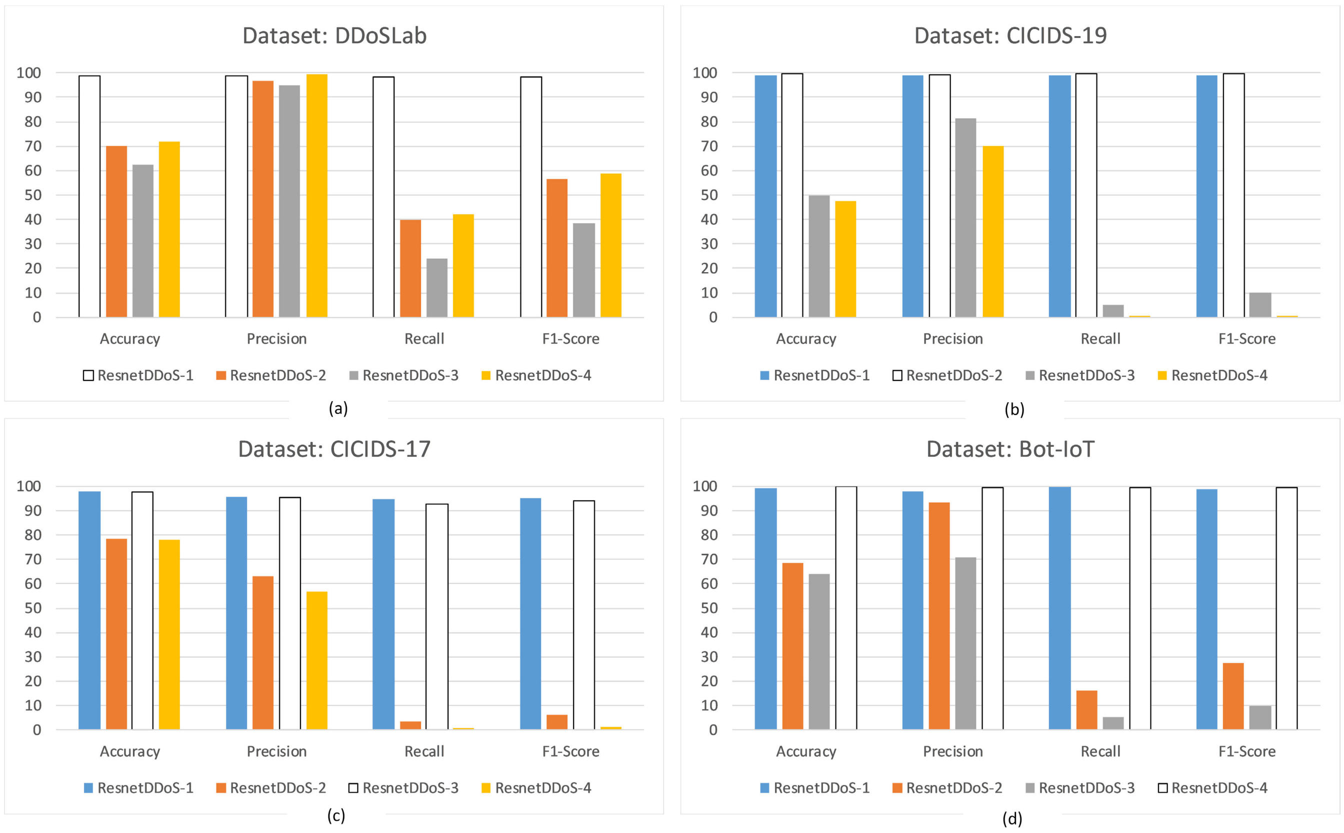

FIGURE 4. **Performance comparison of ResNetDDoS models on individual dataset. (a) Presents the results with the ScanLab dataset. (b) Presents the results with the CICIDS-19 dataset. (c) Presents the results with the CICIDS-17. (d) Presents the results with Bot-IoT dataset.**

model (obtained after training the ResNet-18 [23] model over the DDoSLab dataset) over the test-set of CICIDS-19 [55], CICIDS-17 [56] and Bot-IoT [38] datasets. The experimental results manifest that the ResNetDDoS-1 model performed predominantly well over all three datasets for correctly classifying the normal and DDoS attack traffic. On average, the ResNetDDoS-1 model manifested 98.70% accuracy, 97.53% precision, 97.96% recall, and 97.74% f1-score which is equivalent to its performance when trained and tested over the similar dataset.

Likewise, in the second experiment, we tested the performance of the ResNetDDoS-2 model over the test-set of ScanLab, CICIDS-17 [56] and Bot-IoT [38] dataset. The experimental results showed that on average, the ResNetScan-2 model resulted in 72.40% accuracy, 84.41% precision, 19.80% recall, and 30.13% f1-score as shown in Table 10. As a whole, the average performance of the ResNetScan-2 model over the other DDoS datasets decreased by 27.09%, 14.99%, 79.83%, and 69.39% in case of accuracy, precision, recall, and f1-score respectively. In summary, the performance of the ResNetScan-2 model was significantly downgraded as compared to its performance when trained and tested over a similar dataset.

In the same way, in the third experiment, we tested the performance of the ResNetDDoS-3 model over the test-set of ScanLab, CICIDS-19 [55] and Bot-IoT [38] dataset. The experimental results illustrate that on average, the ResNetDDoS-3 model demonstrated 58.77% accuracy, 82.28% precision, 11.53% recall, and 19.36% f1-score. These results indicate that the ResNetDDoS-3 fairly detected the normal traffic samples, however, it did not fairly detect the DDoS traffic due to which the average f1-score drastically reduced as compared to average precision.

Finally, in the fourth experiment, we tested the performance of the ResNetDDoS-4 model over the test-set of ScanLab, CICIDS-17 [56] and CICIDS-19 [55] dataset. The experimental results summarized in Table 10 present that the ResNetDDoS-4 model illustrated 65.81% accuracy, 75.50% precision, 14.29% recall, and 20.25% f1-score. These results also present that the ResNetDDoS-4 model insufficiently detected the DDoS attack samples, owing to which the average F1-score crucially decreased as compared to average precision.

Fig. 4 (a)-(d) compares performance of all ResNetDDoS models with respect to each individual dataset. The white bars with black boundary line in Fig. 4 (a)-(d), presents the performance of the ResNetDDoS model which is trained and tested over a similar dataset while the other bars present the performance of other ResNetDDoS models on each dataset. From Fig. 4 (a)-(d), it can be noticed that the highest performance is achieved when a ResNetDDoS model is tested over the test-set of the similar dataset on which it was trained. However, if we further observe the Fig. 4 (a)-(d), it can be inferred that the proposed ResNetDDoS model,

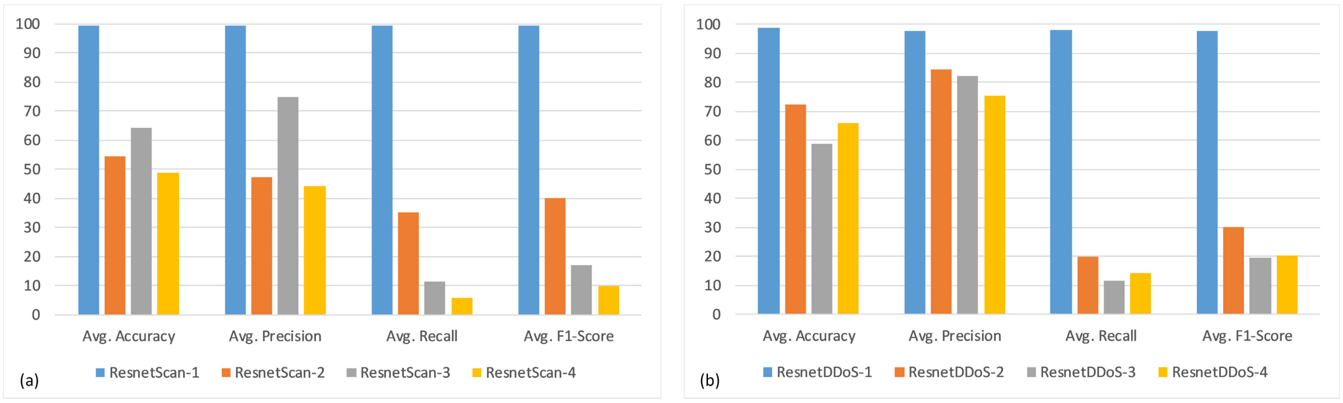

FIGURE 5. Average performance comparison of all (a) ResNetScan and (b) ResNetDDoS models when tested over other datasets that were not used while training.

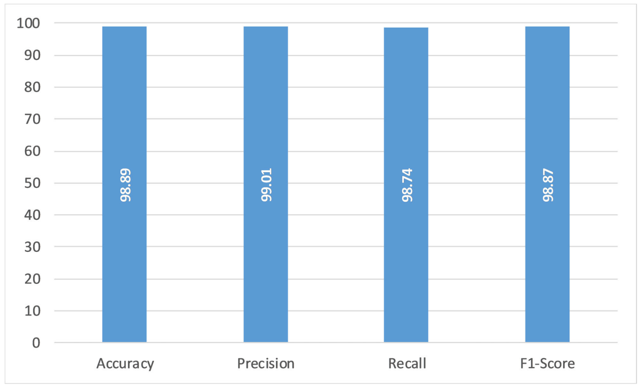

FIGURE 6. Overall performance of the proposed Two-Fold machine learning approach to prevent and detect IoT botnet attacks.

i.e., ResNetDDoS-1 model accomplished the second-highest performance scores and outperformed all other ResNetDDoS models with remarkable accuracy, precision, recall, and f1-score. Thus, the experimental results prove that the proposed ResNetDDoS-1 model outperformed all other ResNetDDoS models when each ResNetDDoS model is tested over the dataset on which it was not trained.

The experimental results of all the above experiments performed in four test scenarios reveal that all the ResNetScan and ResNetDDoS models efficiently detect the scan and DDoS attack when they are tested over the test-set of a similar dataset on which they were trained. However, the performance of all ResNetScan and ResNetDDoS models except ResNetScan-1 and ResNetDDoS-1 model crucially reduced when these models are tested over the test-set of other datasets on which they were not trained. Fig. 5 (a)-(b) illustrates the average performance of all ResNetScan and ResNetDDoS models over the test-set of other datasets on which they were not trained. It can be easily perceived that the average performance of both the proposed ResNetScan-1 and ResNetDDoS-1 models persisted highest as compared to all other models. Hence, the proposed ResNetScan-1 and ResNetDDoS-1 models outperformed all others for detecting scan and DDoS attacks. Overall, the proposed two-fold approach manifested 98.89% accuracy, 99.01% precision, 98.74% recall, and 98.87% f1-score to prevent and detect IoT botnet attacks as displayed in Fig. 6. The testing results and comparative analysis of the proposed two-fold approach on both self-generated datasets and three publicly-available datasets affirm that the proposed two-fold approach is efficient and robust to prevent and detect IoT botnet attacks with large attack patterns coverage.

VI. CONCLUSION

In this work, we proposed a two-fold machine learning approach to prevent and detect IoT botnet attacks. In the first fold, we trained a state-of-the-art deep learning model, i.e., ResNet-18 for scanning attack detection, and named it ResNetScan-1 model. While in the second fold, we trained another ResNet-18 model (named as ResNetDDoS-1 model) in order to detect the DDoS attack in case if the scanning detection model fails to prevent a botnet attack. In order to authenticate the performance of the proposed ResNetScan-1 model and ResNetDDoS-1 model, we performed a couple of experiments in which we take the scan and DDoS traffic samples from three publicly-available datasets, trained the ResNet-18 model over these datasets, and saved the resultant ResNetScan and ResNetDDoS models. We then tested each resultant ResNetScan and ResNetDDoS model over the test-set of other datasets on which they were not trained. The experimental results revealed that the performance of all ResNetScan and ResNetDDoS models except the proposed ResNetScan-1 and ResNetDDoS-1 model crucially reduced when tested over the datasets on which they were not trained. Furthermore, the experimental results proved that the proposed ResNetScan-1 and ResNetDDoS-1 models persisted in their performance and outperformed all other models for detecting the scan and DDoS attacks. Hence, the proposed two-fold approach is efficient and robust to prevent and detect IoT botnet attacks with a large attack patterns coverage.

The current work only covers 33 types of scanning and 60 types of DDoS attacks. In future, we aim to cover more scanning and DDoS attacks techniques in order to well train the proposed framework for more efficient prevention and detection of IoT botnet and DDoS attacks. Further, we can deploy the proposed two-fold approach in an IDS to investigate its effectiveness on live network traffic.

ACKNOWLEDGMENT

This work was supported in part by the Centro de Competências em Cloud Computing (C4) under Grant Operação Centro-01-0145-FEDER-000019, in part by the Programa Operacional Regional do Centro (CENTRO 2020) through the Sistema de Apoio à Investigação Cientíca e TecnológicaProgramas Integrados de IC&DT, in part by FCT/MEC through national funds, in part by FEDERPT2020 Partnership Agreement under Project UIDB/50008/2020, and in part by the Al-Khwarizmi Institute of Computer Science (KICS), University of Engineering and Technology Lahore (UET), Lahore, Pakistan. This article is based upon work from COST Action IC1303–AAPELE–Architectures, Algorithms and Protocols for Enhanced Living Environments and COST Action CA16226–SHELD-ON–Indoor living space improvement: Smart Habitat for the Elderly, supported by COST (European Cooperation in Science and Technology) (More information in www.cost.eu).

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 traffic 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. Prokofiev, 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 filtering using a logistic regression model trained by an artificial 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-ddos-incident-report/

[11] *AWS Said it Mitigated a 2.3 Tbps DDoS Attack, Largest Ever*. Accessed: May 3, 2021. [Online]. Available: https://www.zdnet.com/article/aws-said-it-mitigated-a-2-3-tbps-ddos-attack-the-largest-ever/

[12] *Shodan*. Accessed: May 3, 2021. [Online]. Available: https://www.shodan.io/

[13] *Censys*. Accessed: May 3, 2021. [Online]. Available: https://censys.io/

[14] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, ''DDoS in the IoT: Mirai and other botnets,'' *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[15] R. Hallman, J. Bryan, G. Palavicini, J. Divita, and J. Romero-Mariona, ''IoDDoS—The internet of distributed denial of sevice attacks,'' in *Proc. 2nd Int. Conf. Internet Things, Big Data Secur.* Setúbal, Portugal: SciTePress, 2017, pp. 47–58.

[16] H.-T. Nguyen, Q.-D. Ngo, and V.-H. Le, ''A novel graph-based approach for IoT botnet detection,'' *Int. J. Inf. Secur.*, vol. 19, no. 5, pp. 567–577, Oct. 2020.

[17] F. Hussain, S. G. Abbas, G. A. Shah, I. M. Pires, U. U. Fayyaz, F. Shahzad, N. M. Garcia, and E. Zdravevski, ''A framework for malicious traffic detection in IoT healthcare environment,'' *Sensors*, vol. 21, no. 9, p. 3025, Apr. 2021.

[18] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, ''N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders,'' *IEEE Pervasive Comput.*, vol. 17, no. 3, pp. 12–22, Jul./Sep. 2018.

[19] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, ''A taxonomy and survey of intrusion detection system design techniques, network threats and datasets,'' 2018, *arXiv:1806.03517*.

[20] L. Le Jeune, T. Goedemé, and N. Mentens, ''Machine learning for misuse-based network intrusion detection: Overview, unified evaluation and feature choice comparison framework,'' *IEEE Access*, vol. 9, pp. 63995–64015, 2021.

[21] C. Kim, M. Jang, S. Seo, K. Park, and P. Kang, ''Intrusion detection based on sequential information preserving log embedding methods and anomaly detection algorithms,'' *IEEE Access*, vol. 9, pp. 58088–58101, 2021.

[22] V. T. Alaparthy and S. D. Morgera, ''A multi-level intrusion detection system for wireless sensor networks based on immune theory,'' *IEEE Access*, vol. 6, pp. 47364–47373, 2018.

[23] K. He, X. Zhang, S. Ren, and J. Sun, ''Deep residual learning for image recognition,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[24] W. Wang, Y. Shang, Y. He, Y. Li, and J. Liu, ''BotMark: Automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors,'' *Inf. Sci.*, vol. 511, no. 2, pp. 284–296, Feb. 2020.

[25] W. Yassin, R. Abdullah, M. F. Abdollah, Z. Mas'ud, and F. A. Bakhari, ''An IoT botnet prediction model using frequency based dependency graph: Proof-of-concept,'' in *Proc. 7th Int. Conf. Inf. Technol., IoT Smart City*, Dec. 2019, pp. 344–352.

[26] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, ''IoTPOT: A novel honeypot for revealing current IoT threats,'' *J. Inf. Process.*, vol. 24, no. 3, pp. 522–533, 2016.

[27] S. Almutairi, S. Mahfoudh, S. Almutairi, and J. S. Alowibdi, ''Hybrid botnet detection based on host and network analysis,'' *J. Comput. Netw. Commun.*, vol. 2020, pp. 1–16, Jan. 2020.

[28] A. Blaise, M. Bouet, V. Conan, and S. Secci, ''Botnet fingerprinting: A frequency distributions scheme for lightweight bot detection,'' *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 3, pp. 1701–1714, Sep. 2020.

[29] S. García, M. Grill, J. Stiborek, and A. Zunino, ''An empirical comparison of botnet detection methods,'' *Comput. Secur.*, vol. 45, pp. 100–123, Sep. 2014.

[30] Y. N. Soe, Y. Feng, P. I. Santosa, R. Hartanto, and K. Sakurai, ''Machine learning-based IoT-botnet attack detection with sequential architecture,'' *Sensors*, vol. 20, no. 16, p. 4372, Aug. 2020.

[31] S. Sriram, R. Vinayakumar, M. Alazab, and K. Soman, ''Network flow based IoT botnet attack detection using deep learning,'' in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Jul. 2020, pp. 189–194.

[32] B. Nugraha, A. Nambiar, and T. Bauschert, ''Performance evaluation of botnet detection using deep learning techniques,'' in *Proc. 11th Int. Conf. Netw. Future (NoF)*, Oct. 2020, pp. 141–149.

[33] G. De La Torre Parra, P. Rad, K.-K.-R. Choo, and N. Beebe, ''Detecting Internet of Things attacks using distributed deep learning,'' *J. Netw. Comput. Appl.*, vol. 163, Aug. 2020, Art. no. 102662.

[34] A. Pektaş and T. Acarman, ''Botnet detection based on network flow summary and deep learning,'' *Int. J. Netw. Manage.*, vol. 28, no. 6, p. e2039, Nov. 2018.

[35] A. A. Ahmed, W. A. Jabbar, A. S. Sadiq, and H. Patel, ''Deep learning-based classification model for botnet attack detection,'' *J. Ambient Intell. Hum. Comput.*, pp. 1–10, Mar. 2020. [Online]. Available: https://link.springer.com/article/10.1007/s12652-020-01848-9

[36] S. Maeda, A. Kanai, S. Tanimoto, T. Hatashima, and K. Ohkubo, ''A botnet detection method on SDN using deep learning,'' in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2019, pp. 1–6.

[37] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapé, ''A hierarchical hybrid intrusion detection approach in IoT scenarios,'' in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–7.

[38] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, ''Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset,'' *Future Gener. Comput. Syst.*, vol. 100, pp. 779–796, Nov. 2019.

[39] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, ''Kitsune: An ensemble of autoencoders for online network intrusion detection,'' 2018, *arXiv:1802.09089*.

[40] *NMAP*. Accessed: May 3, 2021. [Online]. Available: https://nmap.org/

[41] *ZMap*. Accessed: May 3, 2021. [Online]. Available: https://github.com/zmap/zmap

[42] *MASSCAN*. Accessed: May 3, 2021. [Online]. Available: https://github.com/robertdavidgraham/masscan

[43] *OpenVAS*. Accessed: May 3, 2021. [Online]. Available: https://www.openvas.org/

[44] S. Gaonkar, N. F. Dessai, J. Costa, A. Borkar, S. Aswale, and P. Shetgaonkar, "A survey on botnet detection techniques," in *Proc. Int. Conf. Emerg. Trends Inf. Technol. Eng. (ic-ETITE)*, Feb. 2020, pp. 1–6.

[45] D. Acarali, M. Rajarajan, N. Komninos, and I. Herwono, "Survey of approaches and features for the identification of HTTP-based botnet traffic," *J. Netw. Comput. Appl.*, vol. 76, pp. 1–15, Dec. 2016.

[46] S. Sikkanan and M. Kasthuri, "Denial-of-service and botnet analysis, detection, and mitigation," in *Forensic Investigations and Risk Management in Mobile and Wireless Communications*. Hershey, PA, USA: IGI Global, 2020, pp. 114–151.

[47] M. U. Nisa and K. Kifayat, "Detection of slow port scanning attacks," in *Proc. Int. Conf. Cyber Warfare Secur. (ICCWS)*, Oct. 2020, pp. 1–7.

[48] F. Tang, Y. Kawamoto, N. Kato, K. Yano, and Y. Suzuki, "Probe delay based adaptive port scanning for IoT devices with private IP address behind NAT," *IEEE Netw.*, vol. 34, no. 2, pp. 195–201, Mar. 2020.

[49] C. Yuan, J. Du, M. Yue, and T. Ma, "The design of large scale IP address and port scanning tool," *Sensors*, vol. 20, no. 16, p. 4423, Aug. 2020.

[50] S. Lee, S.-Y. Im, S.-H. Shin, B.-H. Roh, and C. Lee, "Implementation and vulnerability test of stealth port scanning attacks using ZMap of Censys engine," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2016, pp. 681–683.

[51] E. V. Ananin, A. V. Nikishova, and I. S. Kozhevnikova, "Port scanning detection based on anomalies," in *Proc. Dyn. Syst., Mech. Mach. (Dynamics)*, Nov. 2017, pp. 1–5.

[52] *hping3*. Accessed: May 3, 2021. [Online]. Available: https://tools.kali.org/information-gathering/hping3

[53] *DMitry*. Accessed: May 3, 2021. [Online]. Available: https://tools.kali.org/information-gathering/dmitry

[54] L. Chappell, *Wireshark Network Analysis*. San Jose, CA, USA: Podbooks, 2012.

[55] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2019, pp. 1–8.

[56] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. ICISSP*, 2018, pp. 108–116.

[57] *CICFlowmeter*. Accessed: May 3, 2021. [Online]. Available: https://github.com/ahlashkari/CICFlowMeter

[58] *Network Traffic Flow Analyzer*. Accessed: May 3, 2021. [Online]. Available: http://www.netflowmeter.ca/netflowmeter.html

[59] Y. Xiao and X. Xiao, "An intrusion detection system based on a simplified residual network," *Information*, vol. 10, no. 11, p. 356, Nov. 2019.

[60] R. Vishwakarma and A. K. Jain, "A survey of DDoS attacking techniques and defence mechanisms in the IoT network," *Telecommun. Syst.*, vol. 73, no. 1, pp. 3–25, Jan. 2020.

[61] M. M. Salim, S. Rathore, and J. H. Park, "Distributed denial of service attacks and its defenses in IoT: A survey," *J. Supercomput.*, vol. 76, no. 7, pp. 5320–5363, 2020.

[62] M. De Donno, N. Dragoni, A. Giaretta, and A. Spognardi, "DDoS-capable IoT malwares: Comparative analysis and mirai investigation," *Secur. Commun. Netw.*, vol. 2018, pp. 1–30, 2018.

FAISAL HUSSAIN received the B.Sc. and M.Sc. degrees in computer engineering from the University of Engineering and Technology (UET), Taxila, Pakistan, in 2016 and 2018, respectively. He is currently working as a Senior Research Officer with the Al-Khawarizmi Institute of Computer Science (KICS), UET, Lahore, Pakistan. His research interests include the Internet of Things (IoT), the IoT security, wearable sensors, signal processing, human activity and emotion recognition, pervasive and ubiquitous computing, embedded systems, biomedical engineering, machine learning, neural networks, and data analysis.

SYED GHAZANFAR ABBAS received the master's degree from the Department of Computer Science, National University of Computer and Emerging Sciences (FAST), Lahore, Pakistan, in 2019. He is currently serving as a Senior Research Associate and the Team Lead with the IoT Laboratory, Al-Khawarizmi Institute of Computer Sciences (KICS), UET, Lahore. His research interests include the Internet of Things (IoT) and industry 4.0, machine learning, and cyber security.

IVAN MIGUEL PIRES was born in Penha Garcia, Castelo Branco, Portugal, in 1989. He received the B.Sc. degree in computer science and engineering from the Polytechnic Institute of Castelo Branco, Castelo Branco, in 2010, the M.Sc. degree in computer science and engineering from the Universidade da Beira Interior (UBI), Covilhã, Portugal, in 2012, and the European Ph.D. degree in computer science and engineering from UBI.

From February 2019 to September 2019, he frequented a Postdoctoral Fellowship at the Cloud Computing Competence Centre (C4), Universidade da Beira Interior, related to passing cloud. The subject was conveyed to sensors embedded on mobile devices to monitor traffic warning situations by car. From October 2019 to September 2021, he was an Invited Adjunct Professor with the Polytechnic Institute of Viseu, Viseu, Portugal. He is currently an Invited Assistant Professor with the University of Trás-os-Montes e Alto Douro, Vila Real, Portugal. His main research interests include the use of sensors available in off-the-shelf mobile devices for different purposes, including medicine and sports, and the application of data fusion and data classification techniques of the data acquired from the different sensors.

SABEEHA TANVEER received the B.S. degree in computer science from The University of Lahore (UOL), Lahore, Pakistan, in 2018, and the M.S. degree in computer science from the National University of Computer and Emerging Sciences (FAST), Lahore, in 2020.

She is currently working as a Research Officer with the Al-Khawarizmi Institute of Computer Sciences (KICS), UET, Lahore. Her areas of interests include the IoT, the IoT security, machine learning, and deep learning.

UBAID U. FAYYAZ received the B.S. degree in electrical engineering from the University of Engineering and Technology, Lahore, Pakistan, in 2005, and the M.S. and Ph.D. degrees in electrical engineering from the Georgia Institute of Technology, GA, USA, in 2013 and 2016, respectively.

From 2006 to 2009, he worked with the Center for Advance Research in Engineering (CARE), Islamabad, Pakistan, where he was responsible for the algorithm design and FPGA-based implementations of communication systems. He is currently working as an Assistant Professor with the Electrical Engineering Department, University of Engineering and Technology. His current research interests include coding theory, information theory, and signal processing. He was a recipient of the William J. Fulbright, the Water and Power Development Authority Pakistan, and the National Talent Scholarships.

NUNO M. GARCIA received the B.Sc. degree (Hons.) in mathematics/informatics (for five years) and the Ph.D. degree in computer science engineering from the Universidade da Beira Interior (UBI), Covilhã, Portugal, in 2004 and 2008, respectively.

He has been an Invited Associate Professor with Universidade Lusófona de Humanidades e Tecnologias, Lisbon, Portugal, since 2010. He has also been an Associate Professor with Habilitation at the Computer Science Department, UBI, member of the department, since 2012. His main research interests include next-generation networks, predictive algorithms for healthcare and well-being, distributed and cooperative algorithms, and the battle for a free and open internet.

GHALIB A. SHAH received the Ph.D. degree in computer engineering from Middle East Technical University, Ankara, Turkey, in 2007.

He started his academic career at the College of Electrical and Mechanical Engineering, National University of Sciences and Technology, Islamabad, Pakistan, as an Assistant Professor, in 2007, and served for three years. He has also been with the School of Computer Science, Australian National University, as a Visiting Fellow, from 2009 to 2010. Later, he moved to the Center for Advanced Research in Engineering (CARE), Islamabad, as a member of technical staff and led various networking projects for one year. He then joined the Next Generation Wireless Networks Laboratory, Koc University, Istanbul, as a Research Fellow, and produced significant research publications in well reputed conferences and journals. He joined the Al-Khawarizmi Institute of Computer Science (KICS), UET, Lahore, Pakistan, in 2012, and established the Internet of Things (IoT) Laboratory, where he received many research grants from the HEC Pakistan and the ICT Research and Development Fund. Currently, several IoT and cyber-security related projects are in progress at his laboratory. He is currently a Professor and the Sultan Qaboos IT Co-Chair with KICS. He has over 15 years experience in research and development. His research interests include wireless networking protocols for cognitive radio networks, multimedia communication, the IoT, and the IoT security.

FARRUKH SHAHZAD received the B.C.S. degree (Hons.) from Hamdard University, Karachi, Pakistan, in 1999, the M.S. degree in computer engineering (CE) from the University of Engineering and Technology, Taxila, Pakistan, in 2006, and the Ph.D. degree in electrical engineering (EE) from the National University of Computer and Emerging Sciences (FAST-NUCES), Islamabad, Pakistan, in 2014.

In 2000, he joined Elixir Technologies Pvt. Ltd., Pakistan, as a Software Engineer and worked in the areas of enterprise database applications and printing streams-based applications for heavy duty printers. From 2004 to 2008, he worked as the Team Lead and a Software Architect with Interactive Group Pvt. Ltd., Pakistan, in the area of multimedia applications design and development. In 2008, he joined the Next Generation Intelligent Networks Research Center (nexGIN RC), Islamabad, as the Project Manager, a Software Architect, and a Researcher. He is currently working as a Principal Researcher with Ebryx (Pvt.) Ltd., Pakistan, and also a Researcher with the Al-Khawarizmi Institute of Computer Science (KICS), UET, Lahore, Pakistan. His research interests include cyber security, non-signature base, intelligent and self-healing security solutions for smart phones and desktop operating systems, data mining, and evolutionary algorithms.