IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 20 August 2022, accepted 10 September 2022, date of publication 12 September 2022,
date of current version 22 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3206367

RESEARCH ARTICLE

Anomaly Detection Based on CNN and Regularization Techniques Against Zero-Day Attacks in IoT Networks

BELAL IBRAHIM HAIRAB[1], MAHMOUD SAID ELSAYED[2], ANCA D. JURCUT[2], AND MARIANNE A. AZER[1,3]

[1]School of Information Technology and Computer Science, Nile University, Cairo 12566, Egypt
[2]School of Computer Science, University College Dublin, Dublin 4, D04 V1W8 Ireland
[3]National Telecommunication Institute, Cairo 11768, Egypt

Corresponding author: Mahmoud Said Elsayed (mahmoud.abdallah@ucdconnect.ie)

This work was supported by the University College Dublin (UCD), School of Computer Science, Dublin, Ireland.

ABSTRACT The fast expansion of the Internet of Things (IoT) in the technology and communication industries necessitates a continuously updated cyber-security mechanism to keep protecting the systems' users from any possible attack that might target their data and privacy. Botnets pose a severe risk to the IoT, they use malicious nodes in order to compromise other nodes inside the network to launch several types of attacks causing service disruption. Examples of these attacks are Denial of Service (DoS), Distributed Denial of Service (DDoS), Service Scan, and OS Fingerprint. DoS and DDoS attacks are the most severe attacks in IoT launched from Botnets. Where the Botnet commands previously compromised single or multiple nodes in the network to launch network traffic towards a specific node or service. This leads to computational, power, or network bandwidth draining, which causes specific services to shutdown or behave unexpectedly. In this paper, we aim to verify the detection approach reliability when it encounters an attack that it was not trained on before. Therefore, we evaluate the performance of Convolutional Neural Networks (CNN) classifier in order to detect the malicious attack traffic especially the attacks that never reported before in the network i.e. Zero-Day attacks. Different regularization techniques i.e. L1 and L2 have been used to address the problem of overfitting and to control the complexity of the classifier. The experimental results show that using the regularization methods gives a higher performance in all the evaluation metrics compared to the standard CNN model. In addition, the enhanced CNN technique improves the capability of IDSs in detection of unseen intrusion events.

INDEX TERMS Botnet, convolutional neural networks, distributed denial of service, machine learning, IDS, IoT, zero-day attacks.

I. INTRODUCTION

IoT applications have expanded world-wide, offering the users many services to engage in their personal lives to make it easier. Even though IoT applications are very powerful and give high connectivity to the users and their data over regular apps and systems, they are still considered vulnerable to a wide diversity of attacks and threats [1]. These vulnerabilities are caused by the architecture type of the IoT ecosystem that includes heterogeneous layers of communication. In addition, the power consumption constraints force a low rate of computational power that cannot handle the proper cryptographic calculations between the network's nodes [2]. The possible elasticity of the IoT network (i.e. continuous joining and leaving of unknown nodes) could also generate a vulnerability in the context of securing an IoT network. Therefore, the security and privacy of those users' data became the primary concern to avoid any catastrophic data breaches. Figure 1 shows the ecosystem structure of a typical IoT application.

Botnets are networks formed by compromised nodes in the IoT application. A botnet usually starts when a single malicious node joins the application. This could result from

The associate editor coordinating the review of this manuscript and approving it for publication was Chien-Ming Chen.

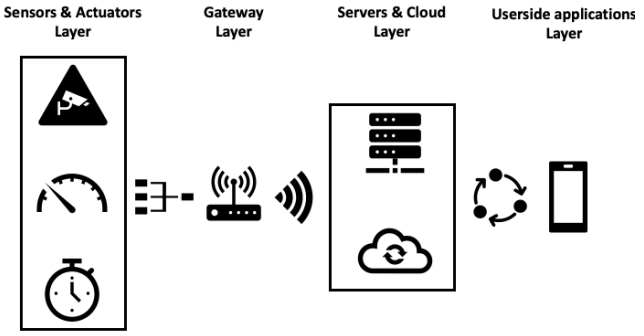


FIGURE 1. Components of an IoT application's ecosystem.

a security vulnerability in the network or a configuration of the IoT application that allows joining stranger nodes. After this malicious node joins the network, it starts to compromise and attack other surrounding nodes to construct a whole network of infected nodes that could launch any attack in the application [3]. The botnet's hierarchy is not flat, there is a botmaster/attacker that exists outside the network and communicates through the internet with the other bots using command and control servers to fire the attacks or any additional actions [4]. Botnet attackers are also have the capability to gather the sensitive Information from the network system, such as Service Scan or OS Fingerprint. The attacker will use this collected information later to start different attacks against the the target victim node/machine.

One of the most common attacks launched by such botnets is the Distributed Denial of Service (DDoS) attack [5]. This targets the availability of certain or multiple resources in the IoT network by flooding them with dummy requests more than they can handle to cut off the desired service from being accessed from the other non-malicious nodes. The effect of this attack could be very catastrophic in some cases, where the accessibility of certain services is very essential to complete a physical-security procedure guaranteed by this IoT application, or any other IoT application related to Smart Cities [6] or Smart Homes. Several types of DDoS attack can be launched in IoT e.g. SYN Flood attack [7] or UDP Flood attack in TCP, and UDP distributions are popular examples of DDoS attacks that target the transport layer in the IoT in OSI model [8]. HTTP flood attack is an example of DDoS attack in the application layer, where the attacker floods the victim with HTTP requests related to the service that the victim's offers in the network.

Networks usually implement Intrusion Detection Systems (IDSs) as a defense against different attacks [9]. IDSs can be categorized based on the detection technique in: (1) Signature-Based, and (2) Anomaly-Based detection. Signature-based relies on comparing a network profile with pre-defined profiles of behaviours to determine if the current activities match an attack profile or not. While the anomaly-based IDSs have a better learning ability to detect the malicious activities. This is done by analyzing the network traffic in a real-time manner to scan for any possible botnet communication, and if detected, it blocks the bots from communicating with each other to prevent launching any attack. Another alternative is by analyzing the network traffic in a time-shifted manner i.e. after an attack started, and depending on the analysis, it blocks the suspicious bots participating in the previous attack. Moreover, some approaches rely on analyzing the power consumption of each IoT device in the network to check for malicious activities. While the real-time approach seems faster with higher protection, it is actually not always reliable to detect unknown attacks. The time-shifted approach has a better chance from time perspective to learn about the traffic patterns and their behaviours and results.

In the last decade, several Machine Learning (ML) and Deep Learning (DL) techniques have been proposed to overcome the challenges of developing an effective intrusion detection system [10], [11].

ML algorithms such as Decision Tree (DT), Logistic Regression (LR), Naive Bayes (NB), ...etc can provide acceptable results when training and testing have the same distribution data. However, when they are tested on a new data distributions (e.g. zero-day attacks scenarios), those classifiers fail to provide a high prediction performance as an expected [5], [12]. This is because classical ML-based methods have low capability to learn the non-linear or the interpretation between the various attacks, especially the attacks that have a high degree of similarity with normal traffic. Furthermore, The classical ML techniques mainly rely on feature engineering to select the best features of the attack classes. However, the best features can be varied from one attack to another. In addition, the feature that can be used for one attack class it is not necessarily to be suitable for another class. The situation which can cause a high false alarm and low detection performance in overall.

On the other hand, Deep Learning (DL) has been wildly used in different application domains (. speech recognition, image processing). It has the capability to extract the intensive features from raw data automatically without prior knowledge [13]. The better performance of DL leads many enterprises, such as Google and Facebook to use it in various applications. Its potential to obtain the hierarchical representation of input data in many applications encourages many researchers to use it in cybersecurity tasks such as anomaly detection. DL can learn the complex and non-linear structure of the input data, in contrast to shallow learners, which require hand-crafted features as input. As a result, we no longer spend time on feature engineering to select appropriate feature sets. However, few studies have investigated the applicability of DL for anomaly detection in IoT networks [14].

Our proposed IDS utilizes the CNN model for effective and early detection of SDN network threats, motivated by the success of the CNN in solving several difficult classification problems. In addition, CNN provides the concept of parameter sharing, which helps significantly to decrease the dimension parameters of the detection model. Although CNN has been employed to identify anomalies [15], [16], it failed to provide acceptable results in discovering abnormalities.

The difference between malicious traffic and normal data is small, and they are pretty similar to one another. However, the CNN should be adapted to identify the small variances between the two borders. To solve this problem, regularization techniques are used in this presented work to reduce overfitting and provide a generalized model that can fit well on unknown data. According to the experimental results, the regularized CNN model outperformed the standard CNN i.e., without using the regularization, and this assisted in improving CNN's ability to identify anomalies in the IoT network.

On the other hand, overfitting is a severe issue in neural networks and often arises when the model performs exceptionally well on training data but exhibits poor intuition on testing. When the model is utilized to evaluate new data points, it fails to generalize the scenario inefficient way. This is during the complexity of the model to learn some noises (such as outliers) of the training data as a specific feature of the input flow. The issue of overfitting can be reduced in a number of ways. One of these solutions is to increase the amount of training data since the model will be forced to generalize to produce good results as it will be impossible to overfit all samples. However, this approach is thought to be costly and limited by the datasets' accessibility, i.e., it can be challenging to find a good dataset, particularly for network traffic. The availability of such datasets can reveal personal information to the public, and the network traffic may contain customer information. Another option is to utilize regularization techniques to apply the penalty for greater weights. As a result, the model's complexity can be reduced. L1 and L2 regularization approaches may be used to improve the neural network model and promote lower weights. Another popular technique to reduce overfitting is using the dropout technique. The concept of dropout is to randomly ignore certain neurons from the network with a probability of P during the training, but all discarding units are used during the testing phase.

The contributions of this paper are as follows:

1) Proposes a regularizer DL model based on the CNN and L1 and L2. Regulator approaches are employed to alleviate the classifier models' overfitting issue. The models based on the regularizer produced the best results compared to classical ML algorithms.
2) Several experiments are carried out to validate the performance of the suggested CNN models against unknown attacks. Furthermore, a small collection of features is used to illustrate how DL models may perform with only a few features. All tests are run on the BOT-IoT dataset, which was created to solve data leakage in IoT networks [17].
3) The efficacy and performance of suggested DL techniques are assessed using a variety of assessment metrics, including accuracy, precision, recall, F1-score, and area under curve. The obtained results show that the CNN model outperforms the standard ML algorithms across all assessment measures.

The remainder of this paper is organized as follows: A background of Botnets, and several ML classifiers are presented in Section II. Section III discusses the used approaches to detect Botnet attacks in the field of IoT. Section IV presents the data preparation process for the classifiers. Experimental results, evaluations, and comparison discussions are presented in section V. Finally, conclusions and future work are presented in section VII.

II. BACKGROUND

In this section, we introduce the background on Botnets and the different ML classifiers.

A. BOTNET TYPES, FAMILIES AND IMPLEMENTATIONS

Malware botnets are known with being high severity attacks that cost the victims huge managerial, and financial losses. Malware developers were able to start the first botnet family that is able to launch a UDP Flood DDoS attack in 2008; this botnet is called Linux/Hydra [18]. Psybot, Chuck Norris, and Tsunami followed in 2009, 2010 respectively [19]. Tsunami was the base for the Bashlitte botnet (2014) then Mirai [20] botnet (2016) which was considered as a whole botnet family due to its multiple variants that were created after.

In general, regardless of the botnet's family or type, a typical botnet has systematic steps in performing its attacks [21]. It starts with developing the malicious software when the botmaster tries to build an unwanted software that may be a virus, worm, spyware, trojans or any other already-existing option if the desired malicious activity is implemented inside this option. In the second stage, the botmaster injects the malware into a victim's device through plenty of available options: spam emails, non-trusted websites, phishing applications, fake cracked versions of expensive software …etc. All these options take advantage of the victim's ignorance when he trusts a spam email sender or website publisher. After the malware has been injected, the botmaster controls the compromised machines through establishing communication sessions with Command-and-Control servers. This is because victims count could reach thousands, which hardens the process of the individual control over all those victims. The botmaster can exploit information from the victims' devices (infected bots) to start his malicious activity like online bank theft, blackmailing or performing a group malicious attack to a new victim such as DDoS attack. A botmaster must keep the communication with the bots as long as possible without being revealed to the infected device's owner to keep it compromised and achieve the malicious target the longest possible.

B. CONVOLUTIONAL NEURAL NETWORK (CNN)

CNNs are structured in different designs compared to traditional neural networks. Each layer in the traditional neural network composites of a set of neurons that are all linked to all neurons in the previous layer. In contrast, instead of entirely coupled neurons with the previous layer, each layer in CNN is only related to a tiny percentage of the neurons. A basic CNN structure consists of three layers: convolution, pooling, and a fully linked layer [22]. A filter or kernel goes

through the input picture and creates a conclusion of an array of integers in the convolutional layer. Multiplying the kernel over a piece of the input produces a single value. By passing the filter through the whole picture creates multiple values, which represent the feature map of the input data. Using several kernels generates various feature maps that reflect various properties of the input tensors. The following equation describes the mathematical description of the convolution layer:

$$M_i = f(M_{i-1} \otimes W_i + b_i) \tag{1}$$

where M_i describes the feature map at layer i and $M_0 = \mathrm{X}$ (input layer), W_i represents the weight vector of the convolution filter at layer i, while the b_i and f represent the bias vector and activation function, respectively. The Rectified Linear Unit (ReLU) activation function is a common non-linear function used in CNN [23]. The potential of CNN returned to the fewer parameters, which are used compared to the traditional neural network since it shares the same weight and bias vector. In addition, unlike traditional machine learning classifiers, it does not require hand-crafted feature extraction. The second layer is the pooling layer down-sampling operation, and aims to reduce the dimensionality of the feature map.

Pooling operations are classified into two types: maximum pooling and average pooling [24]. To obtain the final outputs, the final convolution or polling layer output is processed through one or more fully-connected layers for classification tasks. The final output layer has the same number of nodes or neurons as the number of output classes.

C. OVERFITTING PREVENTION AND REGULARIZATION

Overfitting is one of the severe problems in machine learning techniques, especially in complex models which have a large number of parameters. During the overfitting, the learning model performs well during the training phase, but its performance, unfortunately, is relativity poor on unknown data. One of the common approaches to overcome this problem is to use feature selection techniques to find the most essential features of the input data. These strategies, however, may cause some losses of useful information. An alternate approach is to use regularization methods to regulate model complexity to reduce the pressure on parameter complexity (i.e. weights & biases). Regularization strategies formalize the features by placing a lossless limit on the magnitude of the coefficients. Controlling parameter values can reduce overfitting and improve model performance on unknown data. For example, the L2 regularization technique involves applying a penalty on the square of the weight coefficient values [25]. As a result, the big weights become near zero. We aim to minimize the following cost function during the training process:

$$j(w^1, b^1, \ldots., w^L, b^L) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) \tag{2}$$

where, L is the loss function, w is the wight and b is the bias. Now, using L2 regularization, the loss function will become:

$$j(w^1, b^1, \ldots., w^L, b^L) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L} \left\| w^L \right\|^2 \tag{3}$$

where, λ is a parameter that can be tuned to control the regularization effect. Using large λ, the weight penalty will be large. Similarly, small λ will reduce the effect of regularization. This is trivial, because the cost function must be minimized. By adding the squared norm of the weight matrix and multiplying it by λ, large weights will be driven down in order to minimize the cost function.

D. ML CLASSIFIERS

In general, ML classifiers are widely used in attack detection techniques, as they have the ability to learn the patterns behind the data, which increases the predictions quality [26]. In this section, a brief introduction of popular related classical ML and CNN classifiers is presented.

1) AdaBoost: is a classification algorithm that is based on DT and similar to Random Forest (RF) [27]. AdaBoost creates multiple DTs but with a predefined max depth, rather than the RF that randomly creates DT without unifying the depth of the trees. Order of trees in RF is not important, while it makes a difference in AdaBoost since each learner/stump or single DT uses the decision of the previous stump into account.
2) Logistic Regression (LR) LR is a binary classification algorithm [28] that belongs to the ML set. It depends on creating an s-shaped curve according to a specific feature value. This s-shaped curve differentiates between two binary classes of data; it also helps to assess whether a feature is useful for the prediction process or not.
3) Naïve Bayes (NB): NB [29] is a ML classification algorithm. It mainly depends on calculating the probabilities (also known as likelihoods for discrete data) of the training data features. It also depends on calculating each class's probability in the training data.
   For example, if there is training data that contains 20 records of class A, and 30 records of class B; the classifier calculates the probability of each class as follows:

$$\rho(A) = \frac{20}{30+20} = 0.4$$

$$\rho(B) = \frac{30}{30+20} = 0.6$$

   Then, through the training process, it defines the probability for each feature in the data waiting for the testing phase. In the testing phase, the classifier multiplies the calculated probabilities of the features exist in the record needs to be classified times the $\rho(A)$ probability once, and times the $\rho(B)$. Each result is considered as

the score or probability of the record belong to the class, and the record considered belong to the class that gets higher score.

NB is known for a disadvantage which is if the tested record has a feature that doesn't exist at all in a specific class, the belonging score for this class is always 0; no matter what are the probability values for the other features.

III. RELATED WORK

The detection of Botnet attacks can be done using different approaches according to the used technology during the detection. In this section, we present different Botnet attack mitigation techniques proposed in the literature.

A. STATISTICAL AND ANALYTICAL METHODS

A statistical and analytical approach to detect Botnet attacks in IoT was presented in [30]. It uses a hybrid scheme of Intrusion Detection System (IDS) [31] with both types: signature-based and anomaly-based, integrated with a mitigation framework. The fog layer [32]of the system, encapsulates all the previous components. This fog layer contributes in shifting the load of detection and monitoring required resources outside the IoT devices layer to an intermediate layer between the IoT devices and the endpoint cloud layer. This was done in order to make the best usage of the limited available resources inside the devices. The study conducted the experiment using the Bot-IoT dataset [17] using multiple ML algorithms such as DT, KNN, NB, GB, and XGB in binary and multiple-classification modes. The results showed multiple observations: The first is that signature-based detection is much faster than anomaly-based detection methods, while the second is the superiority of XGBoost and DT over other algorithms looking at their evaluation metrics. Finally, another observation is that algorithms perform better in binary classification mode than the multi-classification mode.

Another type of detection of attacks from Botnets, is called Security Information and Event Management (SIEM) based detection. In [33], the authors suggested a detection system that relies on input from IDS, firewalls, and OS networking logs …etc. After the system gets the desired input, it starts to analyse, and process the received logs files to correlate with any occurred incident. The log files are stored securely in the system, which depends on the large count of the sent packets to a certain target. If an action is detected it alarms the network administrator to reconfigure the firewall of the network by adding new rules that block these packets.

In general, statistical methods are good in lightweight and fast detection cases, which are needed more in IoT networks. However, in some cases, these methods suffer of low prediction quality as they do not get trained to learn the data patterns hidden between the records. They only depend on the static analysis of the data.

B. ML/DL-BASED METHODS

The authors in. [34] provide an evaluation of using a Random Neural Network (RNN) trained only on normal traffic data, in comparison to Long-Short Term Memory (LSTM) to detect SYN flood DDoS attack. The study shows better results in RNN than LSTM. Although RNN has better results than LSTM, it is not considered as sophisticated enough to depend on, as the accuracy was almost 81%.

A new detection classification system based on SVM and CNN ML algorithms was proposed in [35]. The system depends on converting the binary files into visualized images, which size is [64 * 64] pixels, in gray-scale. The CNN and SVM then handle these images to detect whether a file contains a malicious code injected or not. The accuracy of this method is up to 94% in the binary classification case, however, it achieved only 81% in multi classification case. A limitation of this method could be the malicious file structure that could be easily changed from the Botnet attacker without affecting the severity of the file. This would increase the Detection Rate of the system.

An algorithm called Edge2Guard (E2G) was introduced in [36] as a resource-friendly ML algorithm. It was trained and tested over N-BaIoT dataset, which has normal and attack network traffic logs, that were recorded using Mirai and Bashlitte Botnets. They reduced the features from 115 to only two using PCA reduction method. The algorithm depends on creating separate E2G model for each MCU-based IoT device in the system to make the detection more resource-friendly. RF and DT showed the best results among the others with results close to 100%. The disadvantage in this algorithm is that model should be upgraded frequently when necessary, after being trained with data from the developed type of malware action. The suggested update by the authors to use Over-The-Air (OTA) adds difficulties in the deployment process.

In [37], the authors studied an implementation of a new forensic mechanism using ML techniques to detect a malware activity of a Botnet in an IoT network. The study first explains how existing solutions at that time are efficient but have a high false alarm rate. Their proposed scheme is a forensic mechanism that first collects the traffic from the network through tcpdump tool. Then, from the collected traffic, a suitable feature set of the data is extracted using Bro and Argus tools. Afterwards, to start the classification process, the data with the extracted features are exposed to four main algorithms of ML which are: Association Rule Mining (ARM) [38], ANN [39], NB [40], and DT [41]. The target classifiers are then evaluated after conducting the algorithms on the UNSW-B15 dataset [42] using Weka Tool [43] with Accuracy, Overall Success Rate (OSR) and False Alarm Rate (FAR) metrics. According to the authors, the results showed that the DT method [41] was the best among the other methods by achieving 93.23% accuracy and 6.77% FAR. The second best method which was ARM showed nearly twice the value of FAR 13.55% and 86.45% as an accuracy, which are not the best results that can be achieved through DT.

The authors in [44] suggested a detection approach called Particle Deep Framework (PDF). It achieved a high accuracy of almost 99.9% and thus almost 0% FAR. The mentioned method used deep learning with a deep Multi Layer Perceptron (MLP) model alongside with Particle Swarm Optimization (PSO) [45] to spot the hyper-parameters that can significantly increase the Area Under Curve (AUC) while training the model. These parameters were then used to train the model in the last stage. However, this noticeably also increased the required training time for the model [44].

Lue *et al.* [46] proposed a distributed ANN to detect attacks using auto-encoders. Auto-encoders are a special type of neural networks but focuses on establishing or ''encoding'' a different input layer that can be used for better learning processes later using *sigmoid* activation function. The proposed auto-encoder classifier is executed on each sensor of the IoT network, plus, it provides the cloud with all the records it gets as a training data with much higher interval than sensing action.

Jung *et al.* [47] suggested a new CNN [48] based DL model to detect botnet attacks in IoT networks. This model is composed from eight CNN layers based on analyzing the power consumption of the devices in the network. The IoT sensors in the network are: Camera, Router, and Voice Assistant Devices, and the Botnet type was Mirai. The evaluation results done by the authors for their own model showed accuracy values that reaches 96%.

ML-based methods show that they have a better capability of learning about data in order to have a better decision about data classification data in the future [49]. The diversity between the previous studies is among choosing which data to use as the training inputs. Such as the power consumption, network traffic, or converting traffic to visual gray-scale images to have a better use of them. In general, ML-based methods have a deployment challenge in the IoT networks. Because they require high demand of the resources in a device, which is not always the case in IoT. Therefore, this study focuses on developing a ML-based classification system that has several features, such as: avoid over-fitting, detect zero-day attacks and higher ability to learn non-linear relations between data.

Table 1 presents a detailed comparison between the reviewed approaches and our proposed study.

IV. EXPERIMENTAL METHODOLOGY

In this section, we demonstrate the experimental work to detect the botnet attacks. All experiments have been evaluated using Bot-IoT [17] dataset. The proposed model will be trained on the data included both the DoS attack and normal traffic data. We use three different scenarios to test the efficacy of the trained model as the following:

1) **Scenario A:** Testing classifiers on DDoS attack data.
2) **Scenario B:** Testing classifiers on OS Fingerprint attack data.
3) **Scenario C:** Testing classifiers on Service Scan attack data.

We compare the results of all scenarios after evaluating each result using classifiers: LR, NB, AdaBoost, standard CNN, CNN (L1 regularised) and CNN (L2 regularised).

A. DATASET DESCRIPTION

Datasets are important to help verify detection studies and approaches that are ML-based. For botnets, there are several datasets available, but they are not usually sophisticated enough to be used. The limitations of datasets vary from low diversity of available attacks, traffic is generated virtually not from real traffic, duplication of some records, unlabeled data, or low number of features. Given these limitations, we chose the Bot-IoT dataset [17] to train, test, and verify our classifiers in the previous scenarios. This dataset contains multiple attacks data: Service Scanning, Data Theft, Key Logging, OS Fingerprinting, DoS and DDoS. All attacks' data are available in several communication protocols: UDP, HTTP and TCP.

The dataset also contains real and simulated traffic; which is generated using real IoT services and attacking Virtual Machine devices (VMs) that are all connected together using LAN and WAN networks. The IoT traces are also available in the dataset, it also provides 32 features. The generation testbed was designed and implemented in Research Cyber Range lab of UNSW Camberra. The dataset is available in multiple formats: PCAPs, and CSV formats. The Bot-IoT dataset contains around 72 millions records ordered in 74 files with the 35 full-set of features.

B. DATA PREPARATION

The authors of the dataset applied the Information Gain algorithm [50] to select the best effective 10 features of the dataset and wrap them into additional filtered CSV files. However, In this work, we only used 9 features among the selected top 10. Those features are presented in Table 2.

Before executing the experiment, the data has to be prepared to be compatible with the chosen evaluation classifiers. As the Bot-IoT dataset [17] also provides the data in the pre-processed form. This form of data is cleaned from records that does not have accepted values, and the non-numerical values are standardized into numerical values. All of that is after the features has been reduced from 32 features to 10-best features using IG algorithm. This file is available in CSV format which size around 520 MB and contains around 3.7 million records for normal and other 6 types of attacks traffic. In our case, this file represents the starting point of the data preparation procedure. The target of this procedure is to cover the three experimental work scenarios need of data which are four CSV files with the following specifications:

1) Each file has 9 features.
2) The training *.CSV* file has DoS attack data and normal traffic data.
3) The testing *.CSV* contains DDoS attack with normal traffic data as well.
4) The testing *.CSV* contains Service Scan attack data with the normal traffic data.

TABLE 1. Comparison between the reviewed approaches in detecting IoT attacks and our study.

Reference	Approach Description	Advantages	Limitations	Detection Type
Evmorfos et al. [34]	Using RNN to detect SYN flood DDoS Attack in IoT	RNN has better results compared to LSTM	High False Positive rate up to 19.3% Tested only on TCP SYN flood attack.	ML-based
Su et al [35]	Image-based IoT Botnet detection system	Lightweight Model suitable for IoT deployment.	Achieved accuracy ranges between 81% and 94% based on binary or multi classification. ML algorithm depends on data can be easily changed by attacker without affecting the severity of the attack.	ML-based
Sudharsan et at. [36]	E2G MCU based classifier	Resource friendly for MCU-based devices. Trains a separate model for each device.	Difficulties in deployment. As each device has its own model.	ML-based
Al-Duwairi et al. [33]	SIEM-based detection and mitigation of DDoS attacks	Detection system depends on multiple sources of inputs.	Detection based on parsing and analysing without ML or DL algorithms, which might keep it from developing the detection performance.	Signature based
Koroniotis el al. [37]	Framework to detect new malware activities from Botnet	Detection not constrained to one type of an attack, but any Botnet activity in general. Uses 4 classifiers of ML and compares the result.	6.77% FAR for the DT method which had the best result.	ML-based
Koroniotis et al. [44]	Enhancing DL forensic systems using PSO on PDF approaches	Significant increase of the accuracy due to the usage of PSO.	High time for training the model with the optimized features with PSO.	ML-based
Lawal et al. [30]	Hybrid IDS system build over fog computing architecture.	Better network architecture using fog computing.	Multi-classification had less performance than the binary classification.	Hybrid IDS
Luo et al. [46]	Distributed Auto-Encoder ANN	High accuracy results. Resource friendly for each device.	High dependency on devices. High sensitivity to non-clean inputs.	ML-based
Jung et al. [47]	CNN based on Power Consumption	New model that depends on power consumption. Model does not need to read packets or network traffic.	Accuracy values between 88% up to 96%.	ML-based
Our propsed study	Detecting Zero-Day Attacks in IoT using CNN and Classical Machine Learning.	Compared classical ML methods to CNN (DL-based) methods. Used Regularised techniques for CNN to avoid over-fitting among standard CNN. Detecting unknown zero-day attacks gave accuracy up to 98.5% for L2 regularised CNN. L2 CNN had less testing time than L1 and standard CNN.	Used SMOTE to balance data regarding the normal records.	ML-based

TABLE 2. The used features in the proposed study.

Feature Name	Feature Description
seq	Sequence number
stdev	Standard deviation of the captured records
min	Minimum duration of the captured records
mean	Mean duration of the captured records
drate	Number of packets transmitted from destination to source
srate	Number of packets transmitted from source to destination
max	Maximum duration of the captured records
N_IN_Conn_P_SrcIP	Number of inbound connections per source IP.
N_IN_Conn_P_DstIP	Number of inbound connections per destination IP.

5) The testing *.CSV* contains OS Fingerprint attack data with normal traffic data.

The starting point file mentioned earlier, has 3 fields that can be used to filter the file to include only the required type of traffic needed. These fields are *attack*, *category*, and *subcategory*. The field *attack* has digital Boolean values (1 = attack, or 0 = normal). While the *category*, and *subcategory* contains the main attack's category -if attack- and the communication protocol for this record (UDP, HTTP, or TCP) in some attacks like DoS and DDoS, and the exact type of at; if the record does not represent an attack record, the values in both last two columns are *"Normal"*. as follows:

1) Get Specific attack data in addition to the normal records.
2) Extract the 9 features specified in Table 2 in addition to the results column which is *attack*.

In order for any classifier to work properly, it needs to be provided with sufficient data that contains reasonable count of each class (attack and normal) at least in the training phase. Looking into the data distribution of the previous two files in Table 3, it can be clearly noticed that the data distribution is not balanced and there is a shortage in *normal* data.

While Bot-IoT has only 9543 normal records against 72 millions attack record, it can be considered that this dataset is unbalanced.

TABLE 3. Unbalanced data files distribution.

	DoS File	DDoS File	OS Fingerprint	Service Scan
Normal records	500	500	500	500
Total records	1.6 M	1.9 M	25 K	102 K

Unbalanced dataset causes the model bias toward the class that has higher number of records in the training data, which significantly impacts the prediction quality for the minor classes. The reason for this is that in the learning phase, the classifier has a bigger chance to learn about the attack data, while it couldn't learn enough about the normal data. This makes the classifier tend more to predict any giving record as an attack record even if it is a normal record.

Therefore, a tool for over-sampling is required to increase the number of normal records to be reasonably compatible with the attack data in each file. A solution for this might be replicating the normal data records in the file until reaching a balanced state. However, this solution raises another problem which is over-fitting [51], because in this case, the classifier trains over the same records multiple times which results in the memorization of the records instead of learning and understanding. A tool called Synthetic Minority Oversampling Technique (SMOTE) [52] was used to over-sample the minor class (normal data) in our files.

SMOTE increases the number of the minority class without replicating records from it. It does that by plotting the minor class as points in a 2D space and identify the feature vector for each one. Afterwards, the nearest neighbor for each point is defined and generates a new point that relies on the connecting line between the original point and its nearest neighbor. The new point's position depends on a random number between 0 and 1 represents a fraction from the connecting line length.

In our case, we used SMOTE method to increase the normal class records count in our files to achieve the ratio of 4:10 normal:attack as shown in Table 4.

When the data gets ready in the required files with enough instances of all classes, we don't use the whole files' records for the classification. Hence, we specified a constant number of records to be used in each scenario. This is to keep conditions of training and testing procedures consistent among these scenarios for each classifier. We used around 110K of records as a maximum limit (according to availability) for testing, and 350K records for training with random loading process.

C. THE PROPOSED CNN MODEL

In this section, we investigate the potential of DL for anomaly detection and attack classification in the IoT context. The proposed model integrates the CNN architecture with L1 and L2 regularization methods. The CNN is used to extract more detailed representations of the data characteristics. CNN is widely used for image data. However, the network traffic is not an image. So, the first pre-processing step is to convert the network traffic into an image to be suitable for the input of the CNN. The one-dimensional input data with a subset of 9 features is converted to a two-dimensional 3×3 matrix. In addition, since the only gray-scale image is tested in this study, we set the number of channels to 1. After preparing the input data, the CNN model is initialized for training the input data. Fig. 2 illustrates the structure of the CNN model, which is employed in this work. However, the performance of the CNN model significantly relies on the proper selection of the hyper-parameters values. Several factors play an important role to find the best hyper-parameters values, such as the number of convolutional layers, number of filters, size of the filters, stride, padding mechanism, batch size, etc. However, there is not any magic rule to select the best hyper-parameter values in model implementation; rather, it is advised to conduct several tests and trials to identify the best model structure [53]. Therefore, in order to determine the best parameters, we tested a variety of combinations during the implementation phase. The considered hyper-parameters for our CNN model are listed in Table 5.

On the other hand, selecting the number of convolutional layers depends on the characteristics of the input image. Thus, we employed two convolutional layers with dimensions of 32 and 64, respectively. For each layer, a filter of 3×3 size is used with a stride of 1. After the second convolutional layer, a max-pooling layer is used with a size of 2×2 and a stride of 1. While the convolutional layer learns the feature representation of the preceding input, the pooling layer is used to reduce the dimensionality of the feature map. The output of the pooling layer is reshaped through the flatten layer before it passes to another fully connected layer with 128 neurons. We used the non-linear Relu function in all layers before the output. Finally, the classification layer i.e. Softmax, is used in the last layer in order to classify the traffic into normal or attack. We conducted a series of analyses to evaluate the effectiveness of various classifier techniques. Firstly, the output features from the lower convolutional layers are classified using the SoftMax activation function. Different regularization techniques i.e., L1 and L2, have been used to solve the problem of overfitting and to enhance the model performance in zero-day attack detection. We also compared the performance of CNN with three ML techniques, namely LR, NB, and Adaboost. Additionally, two dropout layers were used before the flatten layer and before the fully connected layer to further reduce the likelihood of overfitting. The trained model was tested later on the new portion of the data to show how it performs with data that has not been observed.

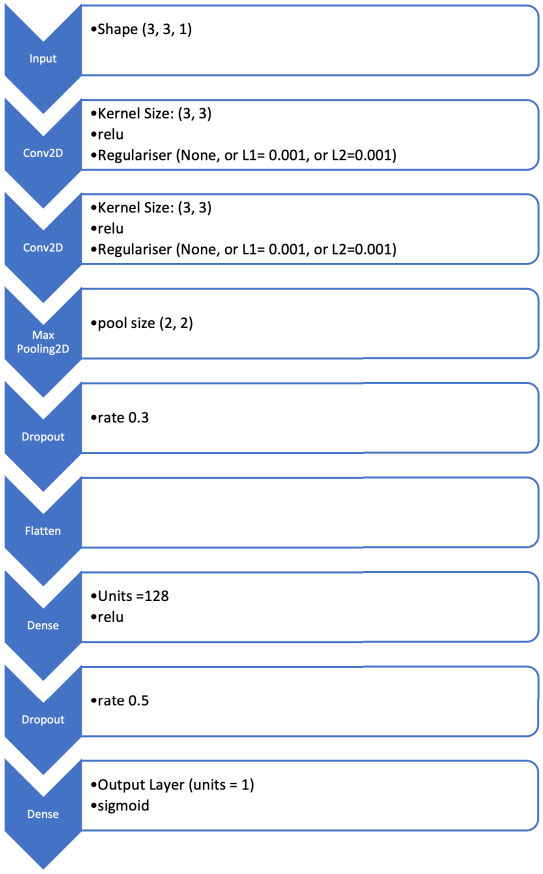


FIGURE 2. The used layers in CNN.

V. EXPERIMENTAL WORK RESULTS

In this section, we define the metrics and measures to be calculated for each classifier based on its results. These metrics help in the process of the classifiers' evaluations and comparison.

A. THE EVALUATION METRICS

To verify and evaluate the suggested deep learning or ML algorithms' performance, several measures are calculated and checked for each scenario among the mentioned scenarios. This helps to have a better understanding and comparison of the performance of each algorithm.

The calculation of the evaluation metrics is done through statistical classification parameters that can be fetched from

TABLE 4. Final data files distribution after using SMOTE to over-sample normal records class.

	DoS File	DDoS File	OS Fingerprint File	Service Scan File
Normal records	660 K	770 K	10 K	41 K
Total records	1.6 M	1.9 M	25 K	102 K

TABLE 5. Used hyper-parameters in CNN.

Hyper-Parameter	Value
Batch Size	32
Number of Epochs	50
Optimizer Method	*adam*

each classifier's confusion matrix [54]. The parameters can be summarized as follows:

1) **True Positive (TP):** A record is considered a TP entry if the algorithm correctly predicted that the studied record belongs to the positive class.
2) **False Positive (FP):** A record is considered a FP entry if the algorithm incorrectly predicted that the studied record belongs to the positive class, while in fact it belongs to the negative class.
3) **False Negative (FN):** A record is considered a FN entry if the algorithm incorrectly predicted that the studied record belongs to the negative class, while in fact it belongs to the positive class.
4) **True Negative (TN):** A record is considered a TN entry if the algorithm correctly predicted that the studied item belongs to the negative class.

After defining the statistical parameters, the evaluation metrics [55] can be better discussed and elaborated:

1) **Accuracy:** This metric defines how much the classification algorithm was correct in its predictions. It is calculated by summing all correct predictions over the total count of predictions.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{4}$$

2) **Precision:** This metric calculates the proportion of correct predictions among the positive classifications.

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

3) **Recall:** (Detection Rate DR), this metric calculates the proportion of actual positive items that were predicted correctly.

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

4) **F1 Score:** This metric provides balance depending on both the Precision and Recall metrics which are used also to calculate it:

$$F1Score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{7}$$

5) **Receiver Operating Characteristic (ROC):** ROC [56] is a curve that illustrates the performance of a classifier by plotting the true positive rate (TPR) against the false positive rate (FPR).

TABLE 6. Specifications of the machine runs the experimental work.

Item	Specification
Workstation	Lenovo Ideapad 500
Operating System	Windows 10 Pro 64-bit
System Version	21H1
OS Build	18362.30
Processor	Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.40 GHz
RAM	8.0 GB.

TABLE 7. The used Python libraries.

Library Name	Usage	Version
Keras	Import DL Layers	2.9.0
Numpy	Arrays Processing	1.20.3
Pandas	Reading and Writing CSV files	1.3.4
SciKit-Learn	ML Classifiers and Metrics	0.24.2
TensorFlow	Prerequisite for Keras	2.9.0

B. ANALYSIS TOOLS

In this work, we executed all experiments on workstation machine that has the following specifications: Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.40 GHz, windows 10 Pro 64-bit operating system with RAM 8.0 GB. We used Python programming language v3.8, from Anaconda with various libraries, the used libraries in our experimental work are presented in Table 7. Anaconda also includes the Python interpreter, many useful libraries, and Spyder IDE. The machine that executed these experiments has specifications illustrated in Table 6.

C. EXPERIMENTAL RESULTS

After defining the statistical parameters and the proper evaluation metrics, the detection algorithms associated with the above scenarios were run to demonstrate and compare the performance of each one. In the following subsections, we represent the experimental results of the three different scenarios.

1) SCENARIO A RESULTS

Table 8 shows the results of CNN model when using the DDoS records in the testing data. We compared the results of the standard CNN with L1 and L2 regularization. For further evaluation, we used three different ML algorithms i.e. LR, NB, Adaboost. It can be noticed that the NB algorithm has the worst evaluation metrics, following by LR. The obtained accuracy from the NB is 75.55%, while the accuracy of LR is 97.55%. It is clear that the standard CNN provides

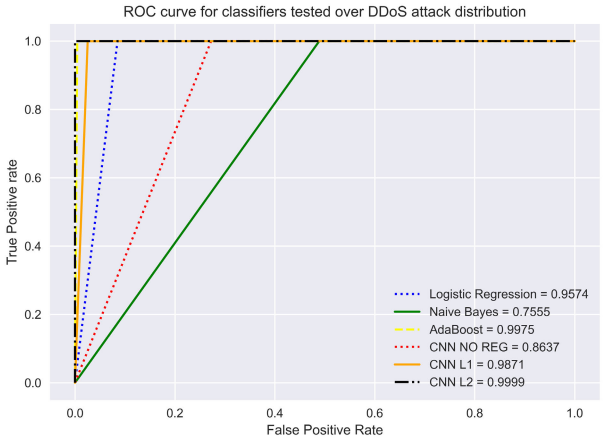


FIGURE 3. ROC for all classifiers for scenario A.

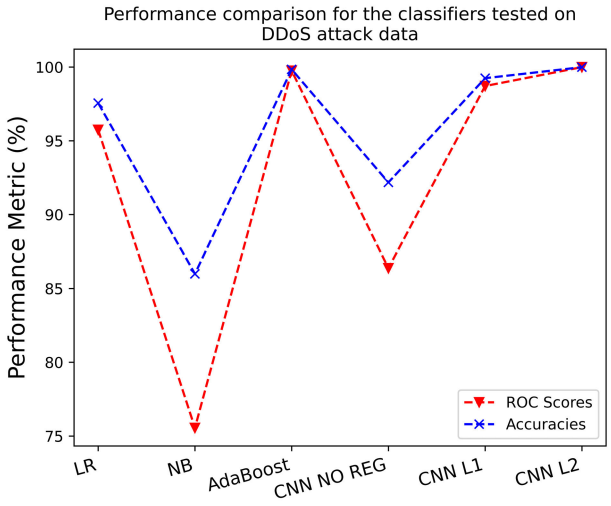


FIGURE 4. Accuracy values and ROC scores of classifiers in scenario A.

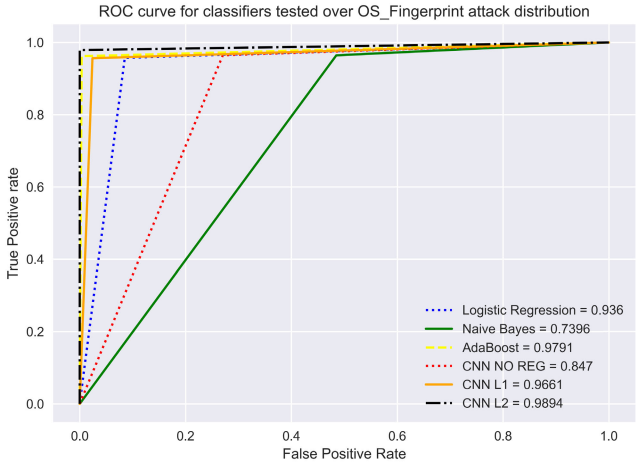


FIGURE 5. ROC for all classifiers for scenario B.

lower evaluation metrics compared to L1 and L2 regularizers. However, the performance of CNN with L2 is much better than CNN with L1. The accuracy of CNN + L2 is reached to 99.98%, while CNN + L1 is 99.24. The reasons behind this results returns to that L2 almost used the square value of magnitude in its calculations, while L1 used the absolute value of magnitude. As a results, L1 assigned a value of 0 for the irrelevant features and it will not never consider them in the calculation. Thus the calculated error can be relativity high. In contrast, the L2 consider the less important features but with less weight values i.e. the value of the weights will never reach to digit zero itself.

For further assess the performance of DL methods, the Receiver operating characteristics (ROC) curve is utilized, as shown in Fig. 3. It represents the trade-off between the true positive and false positive in the graphical structure. The area under the ROC curve demonstrates the overall model performance. The larger area indicates that the model has the capability to differentiate 0s as 0s and 1s as 1s. In contrast, the AUC that was close to zero shows that the model has the lowest level of separability. The experimental results show that the CNN + L2 gives a value of 0.9999, which indicates that 99.99% of positive and negative rates are correctly separated.

While the accuracy values, ROC scores and curves for this scenario for all classifiers are presented in Figures 3 and 4.

From the scenario results analysis and comparison between the classifiers' results, all classifiers achieved reasonable metrics, this might be because of the high similarity between the training and testing attacks, i.e: DoS and DDoS respectively. Furthermore, it can be seen that CNN L2 classifier achieved the best metrics results among other classifiers. Also, the CNN L1 also had a very high metrics but AdaBoost had a bit higher values.

2) SCENARIO B RESULTS

For scenario B, when OS Fingerprint attack is used in the testing phase. Table 9 shows the results of this scenario while the OS Fingerprint attack is considered an unknown attack yet for the classifiers. Same as scenario A, CNN L2 achieved the best results with accuracy up 98.49%. While AdaBoost was the second best with 97.17% accuracy. This is a bit higher over CNN L1 which accuracy was 96.16%. The standard CNN model showed a significant drop in accuracy in comparison to both CNN L1 and L2 with accuracy 89.74% being the second worst classifier after LR.

For the ROC curves and AUC of the CNN models, figure 5 demonstrates that CNN L2 also achieved best AUC with 98.94%, followed by AdaBoost with 97.91%.

Figure 6 visually compares accuracy values and ROC scores between all classifiers. It confirms that the CNN L2 was the best classifier in respect to accuracy and ROC score.

3) SCENARIO C RESULTS

For the last scenario i.e. C, the testing data contains Service Scan attack. In this scenario, the CNN L2 kept the best rank for classification metrics among CNN L1 and other classifiers as well, as presented in table 10. However, unlike scenarios A and B, AdaBoost significantly dropped its

TABLE 8. Evaluation metrics of results of scenario A (In percentage).

	Accuracy	Precision		Recall		F1 Score	
		Normal	Attack	Normal	Attack	Normal	Attack
LR	97.55	99.975	96.69	91.48	99.991	95.542	98.314
NB	75.55	99.98	83.58	51.104	99.997	67.64	91.057
AdaBoost	99.75	99.88	99.81	99.54	99.95	99.71	99.88
CNN	92.17	99.92	90.13	72.75	99.97	84.20	94.80
CNN L1	99.24	99.89	98.98	97.45	99.95	98.65	99.47
CNN L2	**99.98**	**99.96**	**99.99**	**99.98**	**99.98**	**99.97**	**99.99**

TABLE 9. Evaluation metrics of results of scenario B (In percentage).

	Accuracy	Precision		Recall		F1 Score	
		Normal	Attack	Normal	Attack	Normal	Attack
LR	94.51	89.54	96.55	91.47	95.72	90.94	96.14
NB	83.59	85.25	83.24	51.48	96.43	64.20	89.35
AdaBoost	97.17	91.29	99.84	99.62	96.19	95.27	97.98
CNN	89.74	89.18	89.91	72.93	96.46	80.24	93.07
CNN L1	96.19	89.95	99.0	97.57	95.64	93.61	97.29
CNN L2	**98.49**	**95.01**	**99.99**	**99.98**	**97.90**	**97.43**	**98.93**

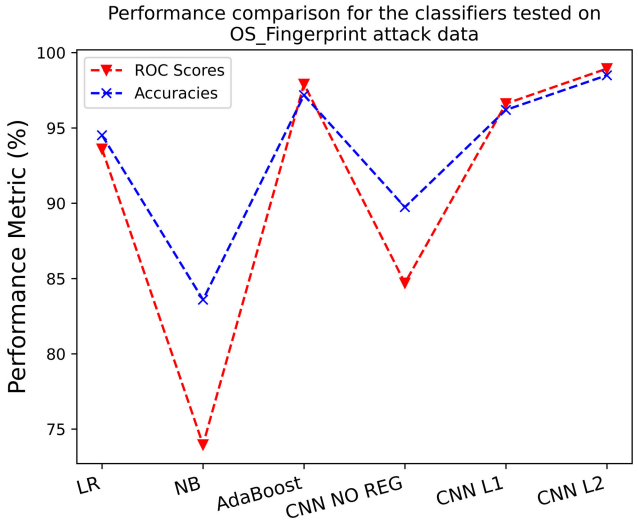


FIGURE 6. Performance metrics comparison for scenario B.

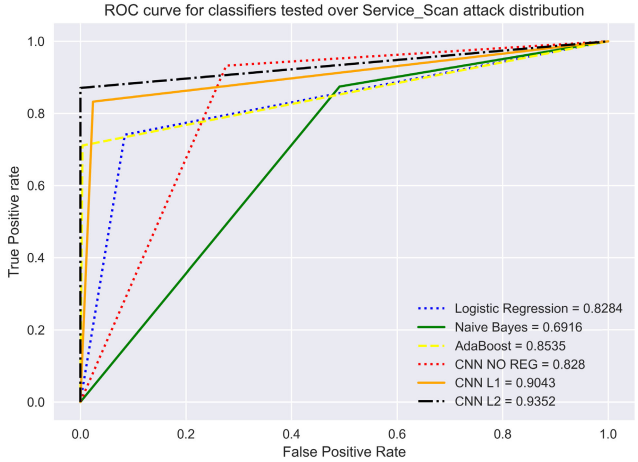


FIGURE 7. ROC for all classifiers for scenario C.

metrics values. While CNN L2 achieved the best accuracy in this scenario: 90.75%, AdaBoost achieved only 79.24%. While CNN L1 and standard CNN were the second and third best classifiers with 87.36% and 87.27% accuracy values respectively.

The results in table 10 seems to split the classifiers into two clusters, DL-based (CNN models) then the classical ML-based models. The accuracy in the DL cluster ranged between 87% and up to 91%. While it ranged in 77% to 80% only in the classical ML-based classifiers. This shows the gained advantage in using CNN models over classical ML-based methods in detecting zero-day attacks. Also, it can be clearly noticed that CNN L2 specifically was ahead of the rest two CNN classifiers, which shows the importance of the L2 regularization method in increasing the CNN's accuracy by avoiding overfitting.

For the ROC scores, figure 7 shows that CNN L2 also was the best classifier from this perspective as well with a value close to 94%. While the second best classifier was CNN L1 with 91% value. On the other hand, NB had the lowest ROC score with only 70%.

Finally, a visual comparison between the classifiers' accuracy values and ROC scores is presented in figure 8. It clearly shows and proves that the L2 regularization method boosted the performance of CNN L2 classifier among other CNN classifiers and ML-based classifiers.

The classifiers were tested on zero-day attacks by steps. It first got trained on DoS attack data, then tested first on DDoS attack data which has a high degree of similarity with the training distributions. As expected in this scenario A, both classical ML-based methods and CNN methods performed well in the testing phase, except for NB. While when we started to test on attacks that has less similarity with the training data (scenarios B, and C), prediction quality had lower values in general comparing to scenario A. AdaBoost performed well on scenario B, but on scenario C it had a significant drop of performance. All in all, the CNN regularised models performed the best among the other used

TABLE 10. Evaluation metrics of results of scenario C (In percentage).

	Accuracy	Precision		Recall		F1 Score	
		Normal	Attack	Normal	Attack	Normal	Attack
LR	79.09	58.58	95.64	91.57	74.10	71.45	83.51
NB	77	61.85	81.65	50.87	87.45	55.83	84.45
AdaBoost	79.24	57.95	99.77	99.59	71.09	73.27	83.02
CNN	87.27	81.07	89.39	72.35	93.24	76.46	91.28
CNN L1	87.36	69.99	98.86	97.60	83.26	81.52	90.39
CNN L2	**90.75**	**75.55**	**99.99**	**99.98**	**87.05**	**86.07**	**93.08**

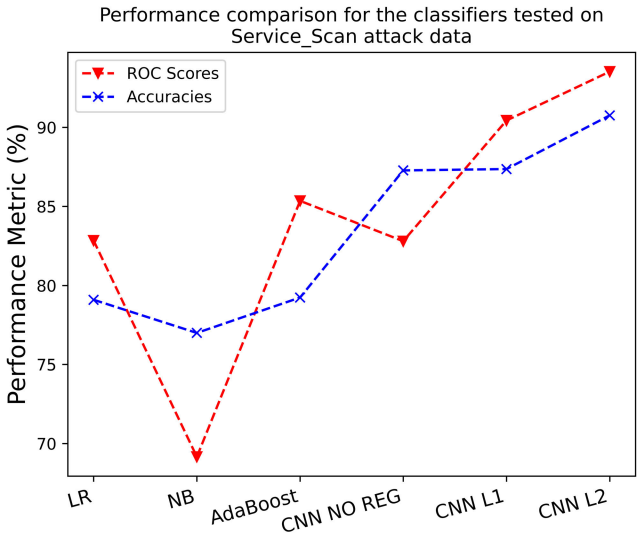


FIGURE 8. Performance metrics comparison for scenario C.

classifiers, especially CNN L2 in scenario C, when it had the best accuracy and ROC score.

VI. DISCUSSION AND LIMITATIONS

Intrusion Detection Systems (IDSs) techniques are considered one of the most critical components of network design. On the other hand, anomaly-based approaches have attracted many academic researchers to identify the attacks on IoT networks in the recent decade. Such approaches potentially have the power to detect new intrusions by monitoring any deviation from the regular traffic pattern. However, one of the major issues impeding the performance of ML/DL models is the problem of overfitting. The model can do quite well during training but fails to represent an acceptable tendency in unseen data. Many factors may cause this problem, including the model's complexity or the insufficient quantity of data utilized to develop an appropriate model. In addition to the aforementioned issue, many existing studies in the intrusion detection field employed the same distribution of testing data similar to those used during the training phase. Evaluating the detection models with such behavior is not a trustworthy strategy for anomaly detection, as any simple algorithm may provide an extremely high accuracy, reached 99% or more. Using the same model with new data (i.e., zero-day attacks) will result in extremely high false rates and poor performance. As a result, the best way to evaluate the performance of intrusion detection models is to observe how they perform with new data that has never been seen before during training. This is what we explored and accomplished in this study. Our proposed model has some drawbacks, which are stated below:

- We trained and evaluated the DL model in offline mode using virtual simulation without implementing a physical IoT networks. However, the detection of attacks online is very important to understand how this IDS can handle the using multi-classification instead of binary classification.
- The adversaries can actively adapt and modify their threat models to learn the decision boundary of the anomaly detector. They aim to compromise the integrity of anomaly detectors by reducing the confidence and modifying the input (an anomalous sample) in order to output (nominal class) by the detector [57]. Therefore, understanding the adversary threat model will help avoid mistakes and reduce the false positive alarms of the anomaly detectors. However, the attack methodology, which adversarial examples reside is beyond the scope of this paper. The interested reader can refer to [57], [58] for more information regarding the general strategies that an attacker can use against any anomaly detector.

VII. CONCLUSION AND FUTURE WORK

In this paper, we evaluated the benefits that are introduced by using CNN models in comparison of the classical ML methods when detecting zero-day attacks. Furthermore, for a deeper evaluation, we used different regularisation methods for the three used CNN models (None, L1, and L2) to help avoid over-fitting and increase the capability of detecting zero-day attacks. For the training process we used 9 features out of 32, those features were selected as the best features using IG methods by the authors of the Bot-IoT dataset. As results, all classifiers including the classical ML-based methods performed well in the first scenario A, as the testing dataset have a high similarity degree with the training dataset. While for zero-day or unseen attacks scenarios (B, and C) classical ML methods performance started to drop down a little. Scenario C showed the most significant performance drop down for all classifiers, however, CNN L2 the best result among the other classifiers with an accuracy close to 91% and ROC score around 94% with a reasonable difference from the other classifiers. In the future, we plan to work on time optimization methods for CNN in order to have faster training times for CNN. In addition, we plan to design other

scenarios having multiple combinations of Botnet attacks with different more diverse datasets. The performance of the various classifiers will be tested when the training and testing data are different.

REFERENCES

[1] N.-F. Polychronou, P.-H. Thevenon, M. Puys, and V. Beroulle, "A comprehensive survey of attacks without physical access targeting hardware vulnerabilities in IoT/IIoT devices, and their detection mechanisms," *ACM Trans. Design Autom. Electron. Syst.*, vol. 27, no. 1, pp. 1–35, Jan. 2022.

[2] P. Anand, Y. Singh, A. Selwal, M. Alazab, S. Tanwar, and N. Kumar, "IoT vulnerability assessment for sustainable computing: Threats, current solutions, and open challenges," *IEEE Access*, vol. 8, pp. 168825–168853, 2020.

[3] N. Koroniotis, "Designing an effective network forensic framework for the investigation of botnets in the Internet of Things," Ph.D. dissertation, School Eng. Inf. Technol., Univ. New South Wales, Sydney, NSW, Australia, 2020.

[4] A. F. Jabbar and I. J. Mohammed, "Development of an optimized botnet detection framework based on filters of features and machine learning classifiers using CICIDS2017 dataset," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 928, no. 3, Nov. 2020, Art. no. 032027.

[5] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "DDoSNet: A deep-learning model for detecting network attacks," in *Proc. IEEE 21st Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Aug. 2020, pp. 391–396.

[6] A. K. M. B. Haque, B. Bhushan, and G. Dhiman, "Conceptualizing smart city applications: Requirements, architecture, security issues, and emerging trends," *Expert Syst.*, vol. 39, no. 5, Jun. 2022, Art. no. e12753.

[7] K. Sonar and H. Upadhyay, "A survey: DDoS attack on Internet of Things," *Int. J. Eng. Res. Develop.*, vol. 10, no. 11, pp. 58–63, 2014.

[8] E. Fraccaroli and D. Quaglia, "Engineering IoT networks," in *Intelligent Internet of Things*. Cham, Switzerland: Springer, 2020, pp. 97–171.

[9] B. I. Hairab and M. A. Azer, "A distributed intrusion detection system for ad hoc networks," in *Proc. 16th Int. Conf. Comput. Eng. Syst. (ICCES)*, Dec. 2021, pp. 1–4.

[10] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Detecting abnormal traffic in large-scale networks," in *Proc. Int. Symp. Netw., Comput. Commun. (ISNCC)*, Oct. 2020, pp. 1–7.

[11] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Network anomaly detection using LSTM based autoencoder," in *Proc. 16th ACM Symp. QoS Secur. Wireless Mobile Netw.*, Nov. 2020, pp. 37–45.

[12] M. S. El Sayed, N.-A. Le-Khac, M. A. Azer, and A. D. Jurcut, "A flow based anomaly detection approach with feature selection method against DDoS attacks in SDNs," *IEEE Trans. Cognit. Commun. Netw.*, early access, Jun. 28, 2022, doi: 10.1109/TCCN.2022.3186331.

[13] M. Abdallah, N. A. L. Khac, H. Jahromi, and A. D. Jurcut, "A hybrid CNN-LSTM based approach for anomaly detection systems in SDNs," in *Proc. 16th Int. Conf. Availability, Rel. Secur.*, Aug. 2021, pp. 1–7.

[14] M. S. ElSayed, N.-A. Le-Khac, M. A. Albahar, and A. Jurcut, "A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique," *J. Netw. Comput. Appl.*, vol. 191, Oct. 2021, Art. no. 103160.

[15] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, "An intrusion detection model based on feature reduction and convolutional neural networks," *IEEE Access*, vol. 7, pp. 42210–42219, 2019.

[16] R.-H. Hwang, M.-C. Peng, C.-W. Huang, P.-C. Lin, and V.-L. Nguyen, "An unsupervised deep learning model for early network traffic anomaly detection," *IEEE Access*, vol. 8, pp. 30387–30399, 2020.

[17] 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. 2018.

[18] P. Beltrán-García, E. Aguirre-Anaya, P. J. Escamilla-Ambrosio, and R. Acosta-Bermejo, "IoT botnets," in *International Congress of Telematics and Computing*. Cham, Switzerland: Springer, 2019, pp. 247–257.

[19] Q.-D. Ngo, H.-T. Nguyen, V.-H. Le, and D.-H. Nguyen, "A survey of IoT malware and detection methods based on static features," *ICT Exp.*, vol. 6, no. 4, pp. 280–286, 2020.

[20] Y. Ji, L. Yao, S. Liu, H. Yao, Q. Ye, and R. Wang, "The study on the botnet and its prevention policies in the Internet of Things," in *Proc. IEEE 22nd Int. Conf. Comput. Supported Cooperat. Work Design ((CSCWD))*, May 2018, pp. 837–842.

[21] H. Cheng and G. R. Regedzai, "A survey on botnet attacks," *American Academic Scientific Research Journal for Engineering, Technology, and Sciences*, vol. 77, no. 1, pp. 76–89, 2021.

[22] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional neural networks for MATLAB," in *Proc. 23rd ACM Int. Conf. Multimedia*, Oct. 2015, pp. 689–692.

[23] D. Zhou, Z. Yan, Y. Fu, and Z. Yao, "A survey on network data collection," *J. Netw. Comput. Appl.*, vol. 116, pp. 9–23, Aug. 2018.

[24] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: An overview and application in radiology," *Insights Imag.*, vol. 9, pp. 611–629, Aug. 2018.

[25] M. S. Elsayed, H. Z. Jahromi, M. M. Nazir, and A. D. Jurcut, "The role of CNN for intrusion detection systems: An improved CNN learning approach for SDNs," in *Proc. Int. Conf. Future Access Enablers Ubiquitous Intell. Infrastructures*. Cham, Switzerland: Springer, 2021, pp. 91–104.

[26] S. H. Haji and S. Y. Ameen, "Attack and anomaly detection in IoT networks using machine learning techniques: A review," *Asian J. Res. Comput. Sci.*, vol. 9, no. 2, pp. 30–46, Jun. 2021.

[27] O. Sagi and L. Rokach, "Ensemble learning: A survey," *WIREs Data Mining Knowl. Discovery*, vol. 8, no. 4, Jul. 2018, Art. no. e1249.

[28] M. Maalouf, "Logistic regression in data analysis: An overview," *Int. J. Data Anal. Techn. Strategies*, vol. 3, no. 3, pp. 281–299, Jul. 2011.

[29] I. Rish, "An empirical study of the naive Bayes classifier," in *Proc. IJCAI Workshop Empirical Methods Artif. Intell.*, vol. 3, no. 22, 2001, pp. 41–46.

[30] M. A. Lawal, R. A. Shaikh, and S. R. Hassan, "An anomaly mitigation framework for IoT using fog computing," *Electron.*, vol. 9, no. 10, p. 1565, 2020.

[31] R. A. Bridges, T. R. Glass-Vanderlan, M. D. Iannacone, M. S. Vincent, and Q. Chen, "A survey of intrusion detection systems leveraging host data," *ACM Comput. Surveys*, vol. 52, no. 6, pp. 1–35, Nov. 2020.

[32] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan, "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE Access*, vol. 6, pp. 47980–48009, 2018.

[33] B. Al-Duwairi, W. Al-Kahla, M. A. AlRefai, Y. Abedalqader, A. Rawash, and R. Fahmawi, "SIEM-based detection and mitigation of IoT-botnet DDoS attacks," *Int. J. Electr. Comput. Eng. (IJECE)*, vol. 10, no. 2, p. 2182, Apr. 2020.

[34] S. Evmorfos, G. Vlachodimitropoulos, N. Bakalos, and E. Gelenbe, "Neural network architectures for the detection of SYN flood attacks in IoT systems," in *Proc. 13th ACM Int. Conf. Pervasive Technol. Rel. Assistive Environ.*, Jun. 2020, pp. 1–4.

[35] J. Su, D. V. Vasconcellos, S. Prasad, D. Sgandurra, Y. Feng, and K. Sakurai, "Lightweight classification of IoT malware based on image recognition," in *Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 2, Jul. 2018, pp. 664–669.

[36] B. Sudharsan, D. Sundaram, P. Patel, J. G. Breslin, and M. I. Ali, "Edge2Guard: Botnet attacks detecting offline models for resource-constrained IoT devices," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops other Affiliated Events (PerCom Workshops)*, Mar. 2021, pp. 680–685.

[37] N. Koroniotis, N. Moustafa, E. Sitnikova, and J. Slay, "Towards developing network forensic mechanism for botnet activities in the IoT based on machine learning techniques," in *Proc. Int. Conf. Mobile Netw. Manage.* Cham, Switzerland: Springer, 2017, pp. 30–44.

[38] A. Telikani, A. H. Gandomi, and A. Shahbahrami, "A survey of evolutionary computation for association rule mining," *Inf. Sci.*, vol. 524, pp. 318–352, Jul. 2020.

[39] F.-L. Fan, J. Xiong, M. Li, and G. Wang, "On interpretability of artificial neural networks: A survey," *IEEE Trans. Radiat. Plasma Med. Sci.*, vol. 5, no. 6, pp. 741–760, Nov. 2021.

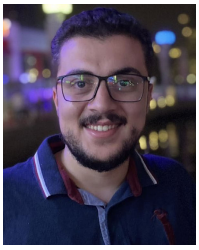
[40] S. Umadevi and K. S. J. Marseline, "A survey on data mining classification algorithms," in *Proc. Int. Conf. Signal Process. Commun. (ICSPC)*, Jul. 2017, pp. 264–268.

[41] D. Kumar, "Decision tree classifier: A detailed survey," *Int. J. Inf. Decis. Sci.*, vol. 12, no. 3, pp. 246–269, 2020.

[42] V. Kanimozhi and P. Jacob, "UNSW-NB15 dataset feature selection and network intrusion detection using deep learning," *Int. J. Recent Technol. Eng.*, vol. 7, no. 5S2, pp. 443–446, 2019.

[43] S. Singhal and M. Jena, "A study on weka tool for data preprocessing, classification and clustering," *Int. J. Innov. Technol. Exploring Eng. (IJItee)*, vol. 2, no. 6, pp. 250–253, 2013.

[44] N. Koroniotis and N. Moustafa, "Enhancing network forensics with particle swarm and deep learning: The particle deep framework," 2020, *arXiv:2005.00722*.
[45] S. Sengupta, S. Basak, and R. A. Peters, II, "Particle swarm optimization: A survey of historical and recent developments with hybridization perspectives," *Mach. Learn. Knowl. Extraction*, vol. 1, no. 1, pp. 157–191, 2019.
[46] T. Luo and S. G. Nagarajan, "Distributed anomaly detection using autoencoder neural networks in WSN for IoT," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
[47] W. Jung, H. Zhao, M. Sun, and G. Zhou, "IoT botnet detection via power consumption modeling," *Smart Health*, vol. 15, Mar. 2020, Art. no. 100103.
[48] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions," *J. Big Data*, vol. 8, no. 1, pp. 1–74, Dec. 2021.
[49] M. S. Elsayed, N.-A. Le-Khac, and A. D. Jurcut, "InSDN: A novel SDN intrusion dataset," *IEEE Access*, vol. 8, pp. 165263–165284, 2020.
[50] T. Z. Win and N. S. M. Kham, "Information gain measured feature selection to reduce high dimensional data," Ph.D. dissertation, Univ. Comput. Stud., MERAL Portal, London, U.K., 2019.
[51] X. Ying, "An overview of overfitting and its solutions," *J. Phys., Conf. Ser.*, vol. 1168, no. 2, 2019, Art. no. 022022.
[52] G. A. Pradipta, R. Wardoyo, A. Musdholifah, I. N. H. Sanjaya, and M. Ismail, "SMOTE for handling imbalanced data problem : A review," in *Proc. 6th Int. Conf. Informat. Comput. (ICIC)*, Nov. 2021, pp. 1–8.
[53] J. Kim, H. Kim, M. Shim, and E. Choi, "CNN-based network intrusion detection against denial-of-service attacks," *Electronics*, vol. 9, p. 916, Jun. 2020.
[54] J. Xu, Y. Zhang, and D. Miao, "Three-way confusion matrix for classification: A measure driven view," *Inf. Sci.*, vol. 507, pp. 772–794, Jan. 2020.
[55] A. B. Sai, A. K. Mohankumar, and M. M. Khapra, "A survey of evaluation metrics used for NLG systems," 2020, *arXiv:2008.12009*.
[56] A. J. Bowers and X. Zhou, "Receiver operating characteristic (ROC) area under the curve (AUC): A diagnostic measure for evaluating the accuracy of predictors of education outcomes," *J. Educ. Students Placed Risk (JESPAR)*, vol. 24, no. 1, pp. 20–46, Jan. 2019.
[57] A. Kuppa, S. Grzonkowski, M. R. Asghar, and N.-A. Le-Khac, "Black box attacks on deep anomaly detectors," in *Proc. 14th Int. Conf. Availability, Rel. Secur.*, Aug. 2019, pp. 1–10.
[58] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognit.*, vol. 84, pp. 317–331, Dec. 2018.



BELAL IBRAHIM HAIRAB received the B.E. degree in systems and biomedical engineering from Cairo University, Cairo, Egypt, in 2018. He is currently pursuing the M.E. degree with the Information Technology and Communications School, Nile University, Cairo. He is also working as a Senior Mobile Software Engineer at Thirdwayv Inc., which is interested in developing secure IoT systems for medical and financial fields.

MAHMOUD SAID ELSAYED received the B.E. degree in electronics and communication engineering from Zagazig University, Zagazig, Egypt, in 2007, the M.E. degree in information security from Nile University, Cairo, Egypt, in 2018, the Ph.D. degree from the School of Computer Science, UCD, Dublin, Ireland, in 2022, the Diploma degree in management of innovation from the DIT Institute, Dublin, and the Diploma degree in networking from the ITI Institute, Cairo. Besides, a set of international professional certificates in the computer networks, security, and IT fields from Cisco, IBM, Huawei, and VMware systems. He has worked for several years in the industry through Huawei and IBM Company in the areas of computer network and security. He is currently a Network and Security Design Engineer in one of the multinational Telecom companies in Ireland. His research interests include but not limited to computer networks, cybersecurity, cyber threat and attacks for android, artificial intelligence, routing protocols, security for the Internet of Things, and software-defined networks.

ANCA D. JURCUT received the B.Sc. degree in computer science and mathematics from the West University of Timisoara, Romania, in 2007, and the Ph.D. degree in security engineering from the University of Limerick (UL), Ireland, in 2013, funded by the Irish Research Council for Science Engineering and Technology. She has been an Assistant Professor with the School of Computer Science, University College Dublin (UCD), Ireland, since 2015. She worked as a Postdoctoral Researcher at UL, a member of the Data Communication Security Laboratory, and as a Software Engineer in IBM, Dublin, Ireland, in the areas of data security and formal verification. Her research interests include security protocols design and analysis, automated techniques for formal verification, network security, attack detection and prevention techniques, security for the Internet of Things, and applications of blockchain for security and privacy. She has several key contributions in research focusing on detection and prevention techniques of attacks over networks, the design and analysis of security protocols, automated techniques for formal verification, and security for mobile edge computing (MEC). For more information visit the link (https://people.ucd.ie/anca.jurcut).

MARIANNE A. AZER received the B.Sc., M.Sc., and Ph.D. degrees from the Electronics and Communications Department, Faculty of Engineering, Cairo University. She is currently an Associate Professor with the National Telecommunication Institute and Nile University, Cairo, Egypt. She is also the Director of the Information Center, National Telecommunication Institute. Her research interests include network security, security in wireless networks, the Internet of Things privacy and security, cloud security, and privacy. She has been a Board Member of the Financial Regulatory Authority, since May 2021. She is a Former Member of the Egyptian Parliament and a Former Advisor of the Minister of Communication and Information Technology for strategic initiatives. She has been the Vice President of Information Systems Audit and Control Association (ISACA) Board, in Egypt, since 2019. Throughout her career, she held several positions, either academic or managerial in several universities and organizations. To mention a few, the Ministry of Communication and Information Technology, the National Telecommunication Institute, Nile University, Cairo University, the American University in Cairo, the French University, the Arab Academy for Science and Technology, and Maritime Transport. She was the President of Information Systems Audit and Control Association (ISACA) Board, in Egypt, in 2018 and 2020, a member of the Global Advisory Board on Emerging Technologies (ISACA) (2020–2021), and also a member of the Global Advisory Board for Facebook Community Leadership Program (2018–2019). She received many awards and recognitions both on the international and national levels. She is a member of international and national organizations in diverse fields, such as telecommunications, politics, women, science, technology, culture, angel investment, and governance.