Hindawi
Journal of Sensors
Volume 2020, Article ID 8828591, 10 pages
https://doi.org/10.1155/2020/8828591

Research Article

A Stacked Deep Learning Approach for IoT Cyberattack Detection

Bandar Alotaibi[1] **and Munif Alotaibi**[2]

[1]*Department of Information Technology, University of Tabuk, Tabuk, Saudi Arabia*
[2]*Department of Computer Science, Shaqra University, Shaqra, Saudi Arabia*

Correspondence should be addressed to Munif Alotaibi; munif@su.edu.sa

Received 6 April 2020; Revised 14 August 2020; Accepted 4 September 2020; Published 18 September 2020

Academic Editor: Kevin Lee

Internet of things (IoT) devices and applications are dramatically increasing worldwide, resulting in more cybersecurity challenges. Among these challenges are malicious activities that target IoT devices and cause serious damage, such as data leakage, phishing and spamming campaigns, distributed denial-of-service (DDoS) attacks, and security breaches. In this paper, a stacked deep learning method is proposed to detect malicious traffic data, particularly malicious attacks targeting IoT devices. The proposed stacked deep learning method is bundled with five pretrained residual networks (ResNets) to deeply learn the characteristics of the suspicious activities and distinguish them from normal traffic. Each pretrained ResNet model consists of 10 residual blocks. We used two large datasets to evaluate the performance of our detection method. We investigated two heterogeneous IoT environments to make our approach deployable in any IoT setting. Our proposed method has the ability to distinguish between benign and malicious traffic data and detect most IoT attacks. The experimental results show that our proposed stacked deep learning method can provide a higher detection rate in real time compared with existing classification techniques.

1. Introduction

Internet of things (IoT) devices are increasingly growing in number and playing very critical roles in our daily lives. IoT devices are behind many popular technology solutions and concepts, such as home automation, autonomous cars, smart cities, Internet of Medical Things, and advanced manufacturing. In IoT, physical devices and everyday objects can be assigned Internet protocol IP addresses and have Internet connectivity. These devices use embedded sensors, processors, and communication hardware to gather, process, and send the data that they capture from their surrounding environments. Thus, they are exposed to many cybersecurity threats [1, 2]. In May 2017, a survey by Synopsys indicated that 67% of manufacturers believe that medical devices are prone to attacks and that medical devices are the most likely to be compromised within 12 months, but only 17% of manufacturers apply procedures to prevent these attacks [3]. A 2020 report by McAfee indicated that cybercriminals are taking advantage of the COVID-19 coronavirus pandemic, leading to a significant increase in several threat categories such as IoT malware, mobile malware, and PowerShell malware. Specifically, McAfee labs perceived 375 cyberthreats per minute during the first quarter of 2020 [4]. Thus, critical systems such as medical IoT devices and healthcare networks have been targeted by cybercriminals as the disease continued to spread [5].

One of the most devastating IoT environment flaws is the lack of security measures. This deficiency makes IoT devices vulnerable to various attacks, such as denial-of-service (DoS), spoofing, data leakage, insecure gateways, and eavesdropping.

These vulnerabilities can cause great harm to hardware and lead to system blackouts, which render services unavailable and even cause physical damage to individuals [6]. Therefore, the security capabilities of IoT devices in IoT technology do not satisfy minimum security requirements, such as CIA (confidentiality, data integrity, and availability). The reasons for the insufficiency of security capabilities of IoT devices are twofold: the heterogeneity of IoT devices with respect to hardware, software, and protocol diversity and limited computational power [7]. In general, it is difficult

Table 1: Description of some IoT terms.

Term	Description
Smart meter	An IoT device that reports information such as gas, water, or energy consumption of a home or organization.
Thermostat	A device that senses a given system's temperature and carries out actions to maintain that temperature. It also permits users to remotely control home heating or air conditioning.
Gateway	A central hub that acts as a bridge that connects sensors or actuators to the Internet.
Switch	A data link layer device that uses a 48-bit identifier (a.k.a. MAC addresses) to convey data.
WiFi access point	A networking device that connects WLAN devices to the wire of the network through a router or a switch.

for an IoT device with low computational power, limited memory size, and restricted battery capability to perform high-complexity security algorithms that demand intensive computation and communication load [8]. Thus, implementing and deploying security measures in IoT environments have been a long-lasting issue [9]. However, security solutions such as NIDSs (network intrusion detection systems) that do not add overhead to IoT devices/environments are effective that they are considered the first line of defense against cyberattacks.

Lately, the main research area within NIDS has been machine and shallow learning methods that are based on well-known algorithms, such as naive Bayes (NB) [10], support vector machine (SVM) [11], and decision tree (DT) [12]. Compared with rule-based expert system NIDS, conventional machine learning-based NIDS approaches have several advantages, such as minimal/no human expert interaction (an expensive process that requires labor-intensive procedures) and reduced errors when constructing the data. Deep learning (a subdomain of machine learning) has been successfully used in many applications, such as computer vision [13], natural language processing [14], and speech recognition [15], and it has achieved state-of-the-art results. Motivated by the success of machine learning-based approaches and the room for improvement in accurately classifying network traffic and identifying attacks, we investigated deep learning to improve the performance of NIDS, especially in terms of accuracy.

1.1. IoT Challenges and Attack Scenarios. Smart things with extensive heterogeneity reside in diversified IoT environments, and they connect to each other via assorted communication links and networking protocols. For instance, devices that reside in smart homes differ from devices that reside on smart grids. Smart home IoT devices include but are not limited to smart appliances, security cameras, baby monitors, connected lighting, smart meters, doorbells, thermostats, solar panels, smoke alarms, and webcams. These devices could reside in one IoT environment setting (i.e., smart home setting), but not in every IoT environment. These devices communicate with each other using a specific

Figure 1: The first attack scenario where the attacker is in control of smart home IoT devices through the used communication technology.

network protocol and connect with the gateway using the same or another network protocol. Devices that reside in different IoT environments such as smart grids can range from power generators, intelligent electronic devices (IEDs), breakers, and substation switches. Some terms used in this subsection and their descriptions are shown in Table 1.

In this research, we investigated two attack scenarios targeting two heterogeneous IoT environments, namely, the smart home and smart grid, to validate our assertion. The smart home environment attack scenario shown in Figure 1 consists of various IoT devices, including doorbells, thermostats, baby monitors, security cameras, and webcams. These devices connect to WiFi access points, which act as coordinators that relay traffic to the wired side of the network through a data link layer device (i.e., a switch). Cyberattacks can be launched using two famous botnets known as Bashlite and Mirai.

The possible attacks launched by Bashlite botnet can be in the form of the following:

(1) Scan: sniffing the network to find vulnerable devices in order to launch other attacks

(2) Junk: launching spam messages

(3) User datagram protocol (UDP): traffic launched using the UDP protocol to flood the network

(4) Transmission control protocol (TCP): traffic launched utilizing the TCP protocol to flood the network

(5) COMBO: launching spam messages and connecting to a given IP address and port number

The possible attacks launched by Mirai are as follows:

(1) Scan: finding vulnerable devices by scanning the network

(2) Ack: launching attacks to flood the network using ack packets

(3) Syn: flood the network by launching syn packets

(4) UDP: launching UDP traffic to flood the network

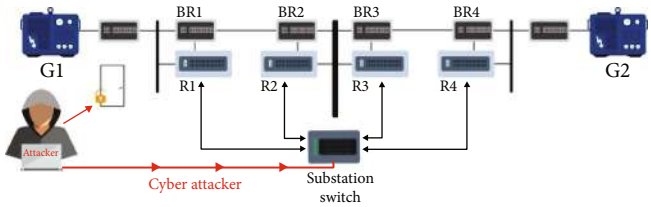

Figure 2: The second attack scenario where the attacker is in control of power system architecture through the substation switch.

(5) UDPPlain: flood the network with UDP traffic using fewer options

The smart grid attack scenario shown in Figure 2 consists of various components that form the configuration of a power system framework. The configuration comprises power generators (denoted by G1 and G2) and intelligent electronic devices (labeled R1 to R4), whose purpose is to switch the breakers on or off. The breakers denoted by BR1–BR4 are connected using two lines: one line connects BR1 and BR2, and the other line connects BR3 and BR4. Additionally, a distance protection scheme that tumbles the breaker to detect faults is used by IEDs. This scheme is used because there is no internal validation mechanism that detects whether the detected faults are faked or valid. Moreover, operators can be used to launch commands to the IEDs to tumble the breakers. The IEDs are connected to the wired side of the network through a substation switch.

The following listed attacks (along with their purposes) can be launched by intruders to harm the IoT environment (i.e., industrial control system):

(1) Remote tripping command injection: the breaker is open as a result of issuing a command (i.e., launching remote tripping command injection) to a relay

(2) Data injection: this attack can be implemented by tweaking values of parameters such as sequence components, current, and voltage to mimic a valid fault, and it affects the operator (the operator encounters a blackout)

(3) Relay setting change: the attacker utilizes the distance protection scheme (used to configure relays) to alter the relay settings in order to block their functions. Thus, the relay will not be able to trip for either a valid command or a proper fault

1.2. Motivation. As we mentioned in Introduction, it is difficult to implement security solutions such as encryption that require decent computation resources in low-constrained IoT devices. The importance of NIDS comes into play here: it can be implemented for powerful computers or even servers at the edge of the IoT environment, and it acts as the first line of defense for all the devices in the IoT environment. However, deploying a consolidated NIDS that works in any IoT environment is a heavy-duty task since these heterogeneous smart things have different characteristics, and the traffic generated by these devices is distinct (i.e., the traffic of legitimate IoT devices in a given IoT environment might be different from traffic generated by legitimate IoT devices in another IoT environment). Motivated by these challenges, we propose an NIDS that can deeply learn the characteristics of IoT devices and detect intruders that intend to harm the IoT environment. We evaluated our proposed method on two heterogeneous IoT environments consisting of heterogeneous IoT devices.

To the best of our knowledge, the architecture of the proposed stacked deep learning approach is novel and improves the performance (in terms of accuracy) of current NIDSs deployed in the IoT paradigms. The contributions of this research paper are the following:

(1) A novel stacked deep learning architecture comprising five pretrained residual network (ResNet) models is proposed to accurately identify IoT cyberattacks

(2) The proposed method achieved promising results on two widely used datasets involving IoT traffic

(3) The proposed method was evaluated on two heterogeneous IoT environments, making the method more thorough and able to detect more attacks than previously proposed methods

The rest of the paper is constructed as follows: Section 2 surveys the related work. The deep learning and stacked generalization ensemble background are introduced in Section 3. Section 4 presents our proposed stacked deep learning approach. Section 5 introduces the empirical evaluation and results. Section 6 analyzes the results. Section 7 concludes our research paper.

2. Related Work

Several methods for the detection of IoT attacks have been previously proposed. For example, a recent research study [16] proposed a feature selection method based on the correntropy and correlation coefficient to evaluate and measure the strength and importance of network traffic features. Then, the AdaBoost ensemble method was used to combine three classification techniques, namely, NB, artificial neural network (ANN), and DT, to detect any malicious events. The proposed method was tested on two datasets, which are designated for IoT traffic and IoT botnet attacks in particular. The proposed method yielded a promising result compared with NIDSs based on conventional algorithms. However, this method is not comprehensive and cannot detect all the attacks that target IoT networks because it was evaluated using DNS and HTTP traffic only, and the generated malicious traffic contained IoT botnet attacks exclusively.

Mirsky et al. [17] proposed an online anomaly detection framework using an unsupervised learning method based on ensemble autoencoders that can efficiently detect abnormal patterns in network traffic. The framework consists of a feature extractor to extract useful features, a feature mapper to map the features to one of the autoencoders, and an anomaly detector to classify and detect any malicious packets in the

network traffic. Some inherent limitations of anomaly detection can markedly affect this approach, taking into consideration the heterogeneity of IoT devices and their unexpected generated traffic. First, attackers can bypass this approach by generating traffic that is similar to normal traffic but includes malicious activities. Second, traffic from legitimate devices that have software defects might be treated as abnormal traffic and thus increase the number of false alerts.

Meidan et al. [18] proposed deep learning autoencoders to detect any attacks launched from IoT bots. In this work, the deep autoencoder was trained on benign instances that exhibited normal behavior only so that it could be trained to reconstruct its inputs. Thus, it succeeded at reconstructing normal observations, but it failed at reconstructing abnormal observations (unknown behaviors). When the reconstruction error accrued, then the given observations were classified as anomalies. A threshold value was used to discriminate between benign and malicious activities. The framework was evaluated on one dataset and was able to detect the abnormal traffic effectively. However, the framework was not evaluated using different IoT settings other than a smart home (i.e., normal traffic was generated through nine smart home devices, such as a doorbell, security camera, and thermostat). Thus, its effectiveness in detecting other IoT attacks and IoT environments has not been investigated. It is an approach designed for a specific purpose: IoT botnet attacks that target smart home devices. This technique might not work in other IoT environments as effectively as it does in smart home settings because of the heterogeneity of IoT environments, in which it is hard to make profiles for the normal traffic of IoT devices. Marcelo et al. [19] designed an adaptive network layer scheme consisting of several building blocks to analyze the network behavior of the Mirai and Bashlite botnet families and develop signatures that can help block the attacks. Prokofiev et al. [20] used the logistic regression model to detect the IoT botnets at early operational steps. These approaches share the same limitation, which is their design to detect IoT malware families (i.e., Mirai and Bashlite). Other attacks that target IoT devices can bypass the NIDS and affect the IoT environment. Derhab et al. [21] proposed an intrusion detection system. Their system utilizes the random subspace ensemble method and k-nearest neighbors (KNN) to detect forged command attacks that are sent to degrade the industrial control process performance. The random subspace (also known as bagging) uses subsets of the features that are taken randomly instead of utilizing the whole feature set. The random subspace is always bundled with a classifier (usually a decision tree classifier) to perform supervised learning. The authors investigated KNN, which achieved promising results, and then, they used random subspace as an ensemble method consisting of KNN to further improve the performance. Although the proposed method yielded promising results, it was only tested on one dataset that was designed for the smart grid environment. When tested in other IoT settings, its effectiveness might deteriorate.

Wang et al. [22] proposed a method based on an artificial recurrent neural network (RNN) known as long short-term memory (LSTM) to detect a data injection attack in the industrial control process. The authors utilized an LSTM characteristic that is capable of predicting temporal sequences and deals with discrete data by constructing the corresponding correlation. The LSTM was trained using normal traffic and tested using the traffic that was generated from the sensors in the testbed. Subsequently, the authors utilized a Euclidean detector (i.e., a detector employing a Euclidean distance mechanism) to measure the deviation of new unseen traffic to detect attacks. The proposed method is a specific-purposed strategy that was introduced to detect data injection attacks in the industrial control process. Thus, its effectiveness in the detection of other attacks that target IoT environments was not investigated. Additionally, the authors only used one dataset, which is not sufficient for evaluating the performance of the proposed method.

Most previously proposed solutions utilized conventional machine learning algorithms to detect IoT cyberattacks. However, these solutions are lacking in terms of accuracy. While there are few solutions based on deep learning, these used off-the-shelf architectures that are designed for solving other problems; hence, they are not accurate enough to detect IoT cyberattacks. Moreover, all of these solutions only seek to address a part of the problem, e.g., detecting smart home cyberattacks (i.e., none of these solutions were evaluated on heterogeneous IoT environments).

3. ResNet and Stacked Ensemble Background

This section discusses the two techniques that we utilized to detect IoT cyberattacks. Deep learning in general and ResNet architecture in particular and the stacked generalization ensemble are introduced in Sections 3.1 and 3.2, respectively.

3.1. ResNet Model Background. Deep learning is considered a subset and evolution of machine learning. Deep learning has reshaped the field of machine learning and has recently gained the attention of the machine learning research community due to its powerful performance. It can extract meaningful data representation and features. Moreover, it can handle a massive amount of data in short span of time, while maintaining outstanding performance. Furthermore, some deep learning algorithms such as the convolutional neural network (CNN) do not require any preprocessing steps. Deep learning algorithms have been able to solve many complicated problems. Particularly, CNN is an advanced deep learning method that has achieved state-of-the-art results in many fields, such as facial, gait, action, and speech recognition; visual object recognition; and object detection. CNN is composed of multiple processing layers, such as convolutional, subsampling, and fully connected layers. The ultimate goal of these layers is to extract and learn the important features and representations of data. Although CNN is making major advances in solving very complicated issues that cannot be solved by other algorithms, optimizing and finding the best CNN model are still complicated issues. In the last decade, researchers have proposed many advanced deep CNN models, such as AlexNet [23], ResNet [24], Xception [25], visual geometry group (VGG) [26], and Inception

[27]. ResNet has shown great performance and has achieved state of the art compared with other CNN architectures.

The ResNet model consists of many ResNet Blocks. A ResNet Block has a special underlying mapping and the "identity shortcut connection" feature, which skips one or more layers.

3.2. Stacked Generalization Ensemble. Stacked generalization [28] or simply stacking was proposed by Wolpert in 1992. A stacked generalization ensemble is a technique that combines multiple base-level classification models via a metaclassifier or metalearner. The outputs of these trained base-level models are combined and used as features to train the meta-classification model. The procedure of stacked generalization is as follows: two disjoint sets of data are split from the training set, the base-level classification models are trained by utilizing the first part, the base-level classification models are tested using the second part, and the predictions generated from the base classification models are used as inputs and the ground truth table as outputs to train the metaclassification model. Note that the training and the testing parts of stacked generalization are similar to cross-validation; however, the base-level classification models are combined (very likely nonlinearly) instead of utilizing a winner-takes-all technique. The accuracy resulting from stacking has been proved to be higher than that of individual classifiers [29].

4. Our Proposed Method

The approach proposed in this paper uses one of the most advanced machine learning algorithms to detect cyberattacks against IoT devices. Our method is based on deep learning. We stacked five deep ResNet models $C^i = c^1, \cdots, c^i$, and the outputs of these models are connected to a new model that takes their output as new training data. The following Algorithm 1 explains the function of our Stacked ResNet Models.

D is the dataset, x^t is the input sample, y^t is the label for that sample, and ^{t} is the index.

In each ResNet model, all 10 ResNet Blocks have two convolutional layers. Given the input X, which is fed into the convolutional layer (L), the operation of the ResNet block is defined as

$$\begin{aligned} X^{i+1} &= L^n(X), \\ X^{i+2} &= L^{n+1}\left(X^{i+1}\right), \\ X^{i+3} &= L^{n+2}\left(X^{i+2}\right) + X, \end{aligned} \tag{1}$$

where ^{n} is the index of the layers. The architecture of our proposed model is shown in Figure 3. Each sub-ResNet model consists of 10 ResNet blocks. Each ResNet is trained individually with training data D. Then, an entirely new model is trained to learn how to best combine the contributions from all five submodels. Thus, as shown in Figure 3, at the testing stage, each of the five submodels receives an input (a single vector) and transforms it through a series of layers. Then, the outputs of all five models are combined to form a new training dataset D^h, which is fed directly to the new meta-algorithm H. The new meta-algorithm consists of two dense layers followed by Softmax to output the class scores. The first dense layer consists of 40 neurons, and the second dense layer consists of 20 neurons. All five ResNet models have the same settings.

Input: Training Data $D = \{x^t, y^t\}$ in
Output: ensemble classifier ***H*** out
Initialisation:
  Step 1: learn base-level classifiers C^i
  for i =1 to 5 **do**
    Learn C^i based on D.
  end for
  Step 2: form new data set of predictions D^h
  for i=1 to 5 **do**
    D^h=$\{x^p, y^t\}$, where $x^p = \{c^1(x^1), \cdots c^i(x^t)\}$
  end for
  Step 3: learn a meta classifier
  Learn H based on D^h
  return H

Algorithm 1: Algorithm for stacked deep ResNet models.

Each convolutional layer has 16 convolutional filters (kernels) that output 16 feature maps. One convolutional layer is used. The size of each convolutional filter is 9, with a stride of 1. The values of these parameters are determined empirically. Moreover, in our case, a 1D convolutional layer is used since the input is one-dimensional. The output of each filter is fed directly to the ReLU activation function, which transforms each value v to the same if it is positive, or it will output zero, as shown in

$$f(x) = \max(0, v), \tag{2}$$

where v is the input value to the activation function. The operation of each convolutional kernel is defined as follows:

$$X^i = R\left(K^n * m^{n-1} + \beta^n\right), \tag{3}$$

where $*$ is the convolution operator. It convolves the kernel K^i with the input image m^{i-1}, adds the bias term β, and then applies the rectifier function R, as defined in Equation (2), to produce a feature map m^i. The output of each filter is padded with zeros so that the output feature map has the same length as the input. At the beginning of training our network, the bias terms in our model are initialized to 0, and the weight matrix of each kernel is initialized using the Glorot (Xavier) uniform initializer [30]. The Xavier initializer draws samples from a uniform distribution within the limits $-l$ and l, where

$$l = \sqrt{6/f_{\text{in}} + f_{\text{out}}}, \tag{4}$$

and where f_{in} is the number of input units in the weight matrix and f_{out} is the number of output units in the weight matrix.After the last ResNet block, average pooling (avg pooling) is used to reduce the size of the data, as shown in Figure 3. We found that average pooling has a slightly better

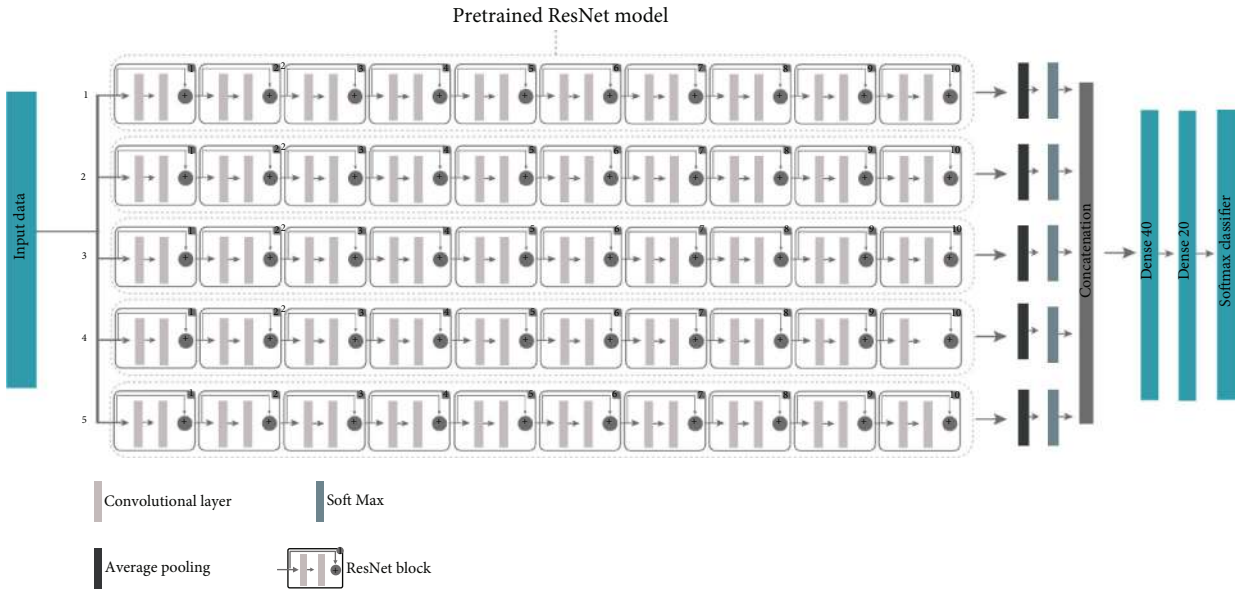

Figure 3: Our stacked deep learning method, which consists of five pretrained residual models. Each pretrained residual model is bundled with 10 residual blocks.

result than max pooling. Our model uses an average pooling size of 2, with a stride of 2, to reduce and downsample the feature maps. The average pooling is defined as

$$y = \frac{1}{k}\sum_{l=1}^{k} ml, \tag{5}$$

where k is the size of the average pooling. The input to average pooling is divided into rectangular pooling regions.

Each region m is downsampled by computing the average of its values.

At the fully connected layer, Softmax is used for classification. Softmax is defined as

$$f_i(s) = \frac{e^{s_i}}{J_{j=1}e^{s_j}} \text{ for } i = 1, \cdots, J, \tag{6}$$

where s is an input vector, and it outputs a vector that has values between zero and one and that has a sum of one.We also used cross entropy during the training which has a loss function and Adam optimizer for the optimization of our model. We used default parameter values, as suggested in the original paper. Thus, we set the values of β_1 to 0.9, β_2 to 0.999, and the learning rate decay to 0. We chose Adam optimizer because it has the advantages of both the RMSProp optimizer and AdaGrad optimizer.

5. Empirical Evaluation and Results

We utilized Keras (the deep learning library for Python programming language) [31] to implement our proposed method. The methods that we compared with our method were implemented using scikit-learn (the Python machine learning library) [32]. All of the experiments were conducted using a laptop running the Windows 10 operating system with 16 GB RAM.

Table 2: The number of instances in each subdataset of the power system dataset.

Dataset	Samples	Dataset	Samples	Dataset	Samples
1	4966	2	5069	3	5415
4	5202	5	5161	6	4967
7	5236	8	5315	9	5340
10	5569	11	5251	12	5224
13	5271	14	5115	15	5276

Table 3: Details of each N-BaIoT subdataset.

Dataset	Benign	Total
Ecobee Thermostat	39,100	835,876
Ennio Doorbell	13,113	355,500
Samsung SNH 1011 N Webcam	52,150	375,222

5.1. Datasets. In our experiments, we endeavored to utilize real-case study scenarios that were carried out in an industrial control system (ICS) (i.e., a smart grid setting) [33] and a smart home setting [18].

5.1.1. Power System Dataset. The first dataset that we used is a power system dataset that is part of the ICS Cyberattack Dataset collection. The power system dataset is divided into three different subdatasets, categorized according to the number of classes that they include (i.e., binary-class subdataset, three-class subdataset, and multiclass subdataset). In this study, we only focused on the binary-class subdataset. The binary-class subdataset has 15 different datasets. Table 2 shows the number of instances in the binary-class

Table 4: The accuracy of our method compared with the tested methods on the power system datasets.

Model	Power system														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Ours	97.7	97.9	97.2	96.0	97.1	96.8	98.3	97.2	96.7	98.0	97.9	98.0	98.5	98.4	96.9
[16]	93.4	89.3	89.8	88.8	32.5	90.2	75.2	74.9	89.1	88.5	89.8	89.6	92.7	90.1	88.5
[21]	90.0	88.5	86.8	84.9	88.6	85.6	88.5	89.2	87.2	88.8	86.7	86.3	88.9	88.1	87.4
RF	93.8	93.1	94.5	93.3	94.0	93.4	94.3	94.0	93.2	94.0	94.3	93.9	96.0	94.8	92.9
LR	87.7	71.0	72.7	67.9	75.0	71.3	77.5	74.3	69.9	73.1	76.3	69.4	78.5	75.0	68.7
DT	92.5	90.0	89.5	89.4	92.3	91.3	89.2	92.3	90.1	87.9	91.2	89.7	92.7	90.6	88.5
LDA	77.6	73.2	72.1	69.0	76.2	72.3	78.7	74.0	70.5	73.2	77.6	69.6	79.1	74.9	70.0
QDA	50.2	51.4	48.8	50.7	44.9	47.5	49.3	47.8	52.8	50.6	43.2	54.3	49.7	46.4	57.3
KNN	89.4	87.4	86.7	85.1	89.2	85.8	88.0	89.2	86.6	88.4	86.0	86.0	88.9	88.1	87.5
NB	25.2	34.1	33.5	39.2	31.5	32.1	29.2	60.8	35.6	33.4	75.8	39.0	27.6	28.9	38.7
XGB	88.3	84.2	88.4	81.7	81.7	82.6	84.7	86.5	84.4	86.4	84.5	80.9	86.3	83.6	83.4
MLP	79.1	69.4	71.1	62.1	71.5	32.0	73.9	28.0	65.7	71.6	76.5	65.2	76.2	28.4	62.7
RSS	93.8	92.1	93.8	92.8	94.2	93.1	92.8	93.9	92.0	92.7	92.0	92.4	95.5	94.3	92.0
GB	90.0	85.7	87.5	82.5	83.5	83.9	85.6	87.3	85.5	87.7	86.0	84.6	87.1	85.7	85.3

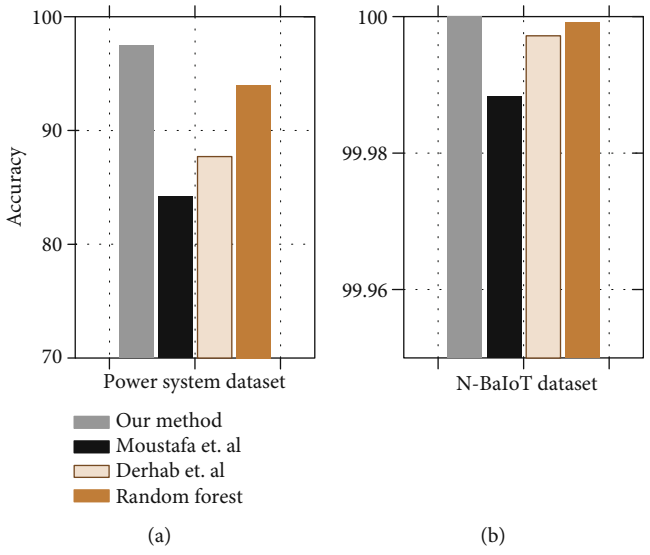

Figure 4: The average accuracy for each of the 15 power system subdatasets is shown in subfigure (a), and the accuracy for three N-BaIoT subdatasets is shown in subfigure (b).

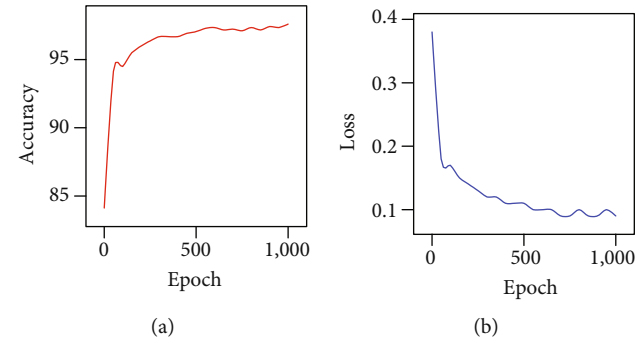

Figure 5: Convergence of the meta-algorithm during the training. Subfigure (a) shows the accuracy against the number of epochs, while subfigure (b) shows the loss against the number of epochs.

subdataset. The number of samples is 78,377, in which the number of normal class samples is 22,714, and the number of abnormal class samples is 55,663. The number of features is 128. The cyberattacks launched to generate abnormal traffic are data injection, remote tripping command injection, and relay setting change. Each sample (instance) in the binary subdataset is classified into a normal event or attack event. The power system and its details are publicly available at https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets.

5.1.2. N-BaIoT: IoT Botnet Attack Dataset. The N-BaIoT dataset [18] consists of nine subdatasets collected from nine IoT devices: Danmini Doorbell, Ecobee Thermostat, Ennio Doorbell, Philips B120N10 Baby Monitor, Philips B120N10 Baby Monitor2, Provision PT 737E Security Camera, Provision PT 838 Security Camera, Samsung SNH 1011 N Webcam, SimpleHome XCS7 1002 WHT Security Camera, and SimpleHome XCS7 1003 WHT Security Camera. Each of the nine sets has malicious data that can be divided into 10 attacks, which are carried by two botnets (i.e., these attacks are integrated into the "attack" class), plus one class of "benign." The details of the three subdatasets of the N-BaIoT dataset that we used in our experiments are shown in Table 3. The N-BaIoT datasets and their details are publicly available at http://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacksN_BaIoT.

5.2. Results. There are 15 power system subdatasets in the binary category. We split each of these 15 into 70% for training and 30% for testing. For example, in the first subdataset, we have 3790 training samples and 1625 testing samples. Each sample is classified as attack or normal. We pretrained each of the five base-level ResNet models

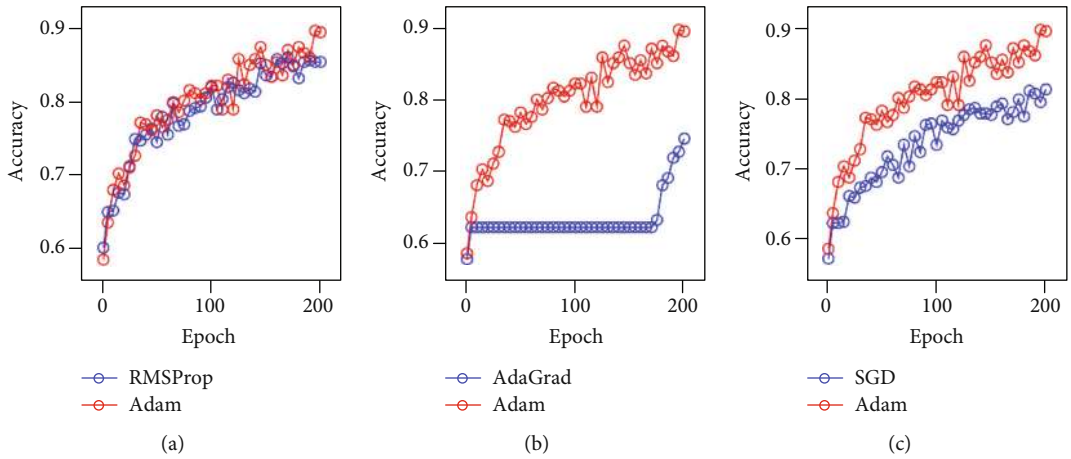

Figure 6: Test accuracy for four adaptive learning rate techniques. Adam optimization versus RMSProp optimizer is shown in (a), (b) shows Adam optimization versus AdaGrad optimizer, and Adam optimizer versus SGD optimizer is shown in (c).

individually using the training set. Using the 30% test dataset, we fed the output of these five ResNets into the metamodel as features. Thus, the metamodel was trained on the outputs of the base-level models.

As described in this subsection, we evaluated the performance of our method and three other similar approaches. Two methods have been proposed recently to detect malicious attacks in the industrial control system environment [21] and smart home setting [16]. We also tested several machine learning algorithms, of which random forest achieved the best accuracy. The most accurate method (i.e., the random forest ensemble method) among the tested methods was included in this study to further evaluate our proposed method. As we mentioned in the previous subsection, the power system dataset and N-BaIoT dataset were utilized to validate our proposed method, particularly binary-class datasets. The first dataset contains 15 subdatasets. For the binary class, our method achieved the best performance in terms of accuracy on all subdatasets. The results of our method when applied to the power system dataset are listed in Table 4.

The average accuracy of our method and the accuracies of the compared methods are shown in Figure 4. The average accuracy was calculated as the accuracy of the tested methods on each subdataset divided by the number of subdatasets. As shown in Figure 4, our method outperformed the other methods when tested on the power system dataset. The average accuracy of our method was 97.5%, which is better than that of the other methods; the second most accurate method was random forests (i.e., its accuracy was 94%).

We also applied our method to the N-BaIoT dataset. This dataset has nine subdatasets. We only utilized three subdatasets to evaluate our proposed method. Similar to the first dataset, we divided the subdatasets into 70% for training and 30% for testing. The results of all methods, including ours, are depicted in Figure 4, which shows that our method achieved the best performance in terms of accuracy on the N-BaIoT dataset. The accuracy of our method and the accuracy of the second-best method (i.e., random forests) are 100% and 99.9991%, respectively.

6. Discussion

As indicated in Results, our proposed method proves to be more accurate than the other algorithms in detecting an IoT cyberattack. All our network parameters, such the number of epochs, the number of filters, ResNet blocks, and the number of neurons in the dense layers, were determined empirically. For example, we empirically determined the number of epochs that would most likely help our model to learn the characteristics of the samples provided in the dataset. During the training process, our model can be optimized very smoothly and reaches the local optima, as shown in Figure 5 (i.e., this shows the convergence of our meta-algorithm). The figure shows both the accuracy and the loss function during the training process of our meta-algorithm.

We also noted that decreasing or increasing the number of ResNet blocks reduced the accuracy. Further, we found that the choice of five ResNet models was optimal. We noted that decreasing the number of epochs above 200 decreases the accuracy, while increasing the number of epochs above 200 does not significantly improve the result and only increases the computation time.

Our method can detect intrusions in real time. The target dataset to measure the detection time for one packet is the power system dataset, and the number of samples on this dataset is 78,377. We split the dataset into 70% for training and 30% for testing for each subdataset. This gives us approximately 23,513 samples for testing and the rest for training. The total testing time for the 23,513 samples is 15,811 ms. The detection time for one packet is the number of samples in the testing set (i.e., 23,513) divided by the total testing time (i.e., 15,811 ms). Thus, the testing time for one packet is around 1.49 ms, which is under the real-time limit.

We also investigated the performance of several optimizers, namely, Adam optimizer [34], the RMSProp optimizer, SGD optimizer, and AdaGrad optimizer [35]. We found that Adam optimizer has the best performance, and it has the advantages of both the RMSProp optimizer and AdaGrad optimizer. Figure 6 shows the accuracy of our model during the training phase when using each of these

optimizers. Moreover, this analysis proves that the Adam optimizer, which is used in our model, is the optimal choice.

7. Conclusions and Future Work

Malicious attacks that target IoT devices are very challenging issues. They target IoT devices and can cause serious problems, such as data leakage, phishing and spamming campaigns, DDoS attacks, and security breaches. In this paper, we propose a stacked deep learning method to detect malicious traffic data, particularly malicious attacks that target IoT devices. This method utilizes a residual block architecture to identify activities that might harm the IoT environment. We investigated several machine learning algorithms and two related work methods to compare with our proposed approach. We evaluated our method using two well-known datasets that are from two heterogeneous IoT environments. Experimental results show that our method is able to deeply learn the characteristics of heterogeneous IoT devices in two different IoT environments. Thus, our proposed method has better predictive performance compared with the related approaches and can achieve good results. In the future work, we plan to investigate transfer learning and several other state-of-the-art pretrained models to improve the performance of IoT IDSs in terms of accuracy and detection time. We will also integrate other heterogeneous IoT environments to prove that IoT network traffic generated by cybercriminals is similar, and a general IDS can be deployed in heterogeneous IoT environments. We will also investigate the impact of deceasing or increasing the number of ResNet blocks on accuracy.

Abbreviations

DDoS: Distributed denial-of-service
ICS: Industrial control system
ReLU: Rectified-linear-unit systems
IoT: Internet of things
VGG: Visual geometry group
RAM: Random access memory
ResNet: Residual network
Xception: Extreme inception
IED: Intelligent electronic devices
NIDS: Network intrusion detection system
CIA: Confidentiality, data integrity, and availability
SVM: Support vector machines
RF: Random forest
NB: Naive Bayes
ANN: Artificial neural network
DT: Decision tree
KNN: *k*-nearest neighbors
RNN: Recurrent neural network
LSTM: Long short-term memory
LR: Logistic regression
LDA: Linear discriminant analysis
QDA: Quadratic discriminant analysis
XGB: eXtreme gradient boosting
MLP: Multilayer perception
RSS: Random subspace
GB: Gradient boosting
DNS: Domain name system
HTTP: Hypertext transfer protocol
WLAN: Wireless local area network
WiFi: Wireless fidelity
MAC: Media access control.

Data Availability

In our experiments, we endeavored to utilize real-case study scenarios that were carried out in an industrial control system (ICS) (i.e., a smart grid setting) [34] and a smart home setting [18]. (1) Power system dataset: the first dataset that we used is a power system dataset that is part of the ICS Cyberattack Dataset collection [34]. The power system and its details are publicly available at https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets. (2) N-BaIoT: IoT Botnet Attack Dataset: the N-BaIoT dataset [18] consists of nine subdatasets collected from nine IoT devices. The N-BaIoT datasets and their details are publicly available at http://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT.

Conflicts of Interest

The authors hereby certify that they have no conflicts of interest. The authors have no affiliation with any organization with a direct or indirect financial interest in the subject matter discussed in the manuscript.

References

[1] E. Anthi, L. Williams, M. Słowińska, G. Theodorakopoulos, and P. Burnap, "A supervised intrusion detection system for smart home IoT devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9042–9053, 2019.

[2] B. Alotaibi, "Utilizing blockchain to overcome cyber security concerns in the internet of things: a review," *IEEE Sensors Journal*, vol. 19, no. 23, pp. 10953–10971, 2019.

[3] "Cybersecurity executive: medical devices a 'bulls-eye' for cyber-attacks," https://www.digitalhealth.net/2017/12/medical-device-functionality-vs-cybersecurity/.

[4] "McAfee labs COVID-19 threats report," July 2020 https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-july-2020.pdf.

[5] H. S. Lallie, L. A. Shepherd, J. R. Nurse et al., "Cyber security in the age of covid-19: a timeline and analysis of cyber-crime and cyber-attacks during the pandemic," 2020, http://arxiv.org/abs/2006.11929.

[6] E. Anthi, A. Javed, O. Rana, and G. Theodorakopoulos, "Secure data sharing and analysis in cloud-based energy management systems," in *Cloud Infrastructures, Services, and IoT Systems for Smart Cities. IISSC 2017, CN4IoT 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 189*, A. Longo, Ed., Springer, Cham, 2017.

[7] J. Frahim, C. Pignataro, J. Apcar, and M. Morrow, *Securing the internet of things: a proposed framework*, *IEEE Cisco White Paper*, 2015.

[8] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, "IoT security techniques based on machine learning: how do IoT devices use AI to enhance security?," *IEEE Signal Processing Magazine*, vol. 35, no. 5, pp. 41–49, 2018.

[9] E. Anthi, S. Ahmad, O. Rana, G. Theodorakopoulos, and P. Burnap, "EclipseIoT: a secure and adaptive hub for the internet of things," *Computers & Security*, vol. 78, pp. 477–490, 2018.

[10] L. Koc, T. A. Mazzuchi, and S. Sarkani, "A network intrusion detection system based on a hidden naïve Bayes multiclass classifier," *Expert Systems with Applications*, vol. 39, no. 18, pp. 13492–13500, 2012.

[11] S. J. Horng, M. Y. Su, Y. H. Chen et al., "A novel intrusion detection system based on hierarchical clustering and support vector machines," *Expert Systems with Applications*, vol. 38, no. 1, pp. 306–313, 2011.

[12] D. Moon, H. Im, I. Kim, and J. H. Park, "DTB-IDS: an intrusion detection system based on decision tree using behavior analysis for preventing APT attacks," *The Journal of Supercomputing*, vol. 73, no. 7, pp. 2881–2895, 2017.

[13] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu, "CNN-RNN: a unified framework for multi-label image classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2285–2294, Las Vegas, NV, USA, 2016.

[14] A. Kumar, O. Irsoy, P. Ondruska et al., "Ask me anything: dynamic memory networks for natural language processing," in *International conference on machine learning*, pp. 1378–1387, New York, NY, USA, 2016.

[15] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: a neural network for large vocabulary conversational speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4960–4964, Shanghai, China, March 2016.

[16] N. Moustafa, B. Turnbull, and K. K. R. Choo, "An ensemble intrusion de- tection technique based on proposed statistical flow features for protecting network traffic of internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4815–4830, 2019.

[17] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," 2018, http://arxiv.org/abs/1802.09089.

[18] Y. Meidan, M. Bohadana, Y. Mathov et al., "N-BaIoT—network- based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[19] J. Ceron, K. Steding-Jessen, C. Hoepers, L. Granville, and C. Margi, "Improving IoT botnet investigation using an adaptive network layer," *Sensors*, vol. 19, no. 3, p. 727, 2019.

[20] A. O. Prokofiev, Y. S. Smirnova, and V. A. Surov, "A method to detect Internet of Things botnets," in *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pp. 105–108, Moscow, Russia, January-Febuary 2018.

[21] A. Derhab, M. Guerroumi, A. Gumaei et al., "Blockchain and random subspace learning-based IDS for SDN-enabled industrial IoT security," *Sensors*, vol. 19, no. 14, p. 3119, 2019.

[22] W. Wang, Y. Xie, L. Ren, X. Zhu, R. Chang, and Q. Yin, "Detection of data injection attack in industrial control system using long short term memory recurrent neural network," in *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp. 2710–2715, Wuhan, China, May-June 2018.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classi- fication with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, Curran Associates, Inc, 2012.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, Las Vegas, NV, USA, 2016.

[25] F. Chollet, "Xception: deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, Honolulu, HI, USA, 2017.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in deep convolutional Networks for Visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[27] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, Boston, MA, USA, 2015.

[28] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[29] M. Sewell, "Ensemble learning," in *The Handbook of Brain Theory and Neural Networks*, The MIT Press, Cambridge, MA, USA, 2008.

[30] X. Glorot and Y. Bengio, "Understanding the difficulty of train- ing deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, Sardinia, Italy, 2010.

[31] F. Chollet, "Keras," 2015, https://keras.io.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort et al., "Scikit-learn: machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[33] R. C. B. Hink, J. M. Beaver, M. A. Buckner, T. Morris, U. Adhikari, and S. Pan, "Machine learning for power system disturbance and cyber-attack discrimination," in *2014 7th International Symposium on Resilient Control Systems (ISRCS)*, pp. 1–8, Denver, CO, USA, August 2014.

[34] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *3rd International Conference on Learning Representations (ICLR2015)*, San Diego, 2015.

[35] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. 7, 2011.