electronics

Article

Anomaly Detection of Zero-Day Attacks Based on CNN and Regularization Techniques

Belal Ibrahim Hairab [1], Heba K. Aslan [2,3], Mahmoud Said Elsayed [3,*], Anca D. Jurcut [4] and Marianne A. Azer [1,5]

1 School of Information Technology and Computer Science, Nile University, Cairo 12677, Egypt
2 Informatics Department, Electronics Research Institute, Cairo 12622, Egypt
3 Center of Informatics Science, Faculty of Information Technology and Computer Science, Nile University, Giza 12588, Egypt
4 School of Computer Science, University College Dublin, 7777 Belfield, Ireland
5 National Telecommunication Institute, Cairo 11765, Egypt
* Correspondence: mahmoud.abdallah@ucdconnect.ie

Abstract: The rapid development of cyberattacks in the field of the Internet of things (IoT) introduces new security challenges regarding zero-day attacks. Intrusion-detection systems (IDS) are usually trained on specific attacks to protect the IoT application, but the attacks that are yet unknown for IDS (i.e., zero-day attacks) still represent challenges and concerns regarding users' data privacy and security in those applications. Anomaly-detection methods usually depend on machine learning (ML)-based methods. Under the ML umbrella are classical ML-based methods, which are known to have low prediction quality and detection rates with regard to data that it has not yet been trained on. DL-based methods, especially convolutional neural networks (CNNs) with regularization methods, address this issue and give a better prediction quality with unknown data and avoid overfitting. In this paper, we evaluate and prove that the CNNs have a better ability to detect zero-day attacks, which are generated from nonbot attackers, compared to classical ML. We use classical ML, normal, and regularized CNN classifiers (L1, and L2 regularized). The training data consists of normal traffic data, and DDoS attack data, as it is the most common attack in the IoT. In order to give the full picture of this evaluation, the testing phase of those classifiers will include two scenarios, each having data with different attack distribution. One of these is the backdoor attack, and the other is the scanning attack. The results of the testing proves that the regularized CNN classifiers still perform better than the classical ML-based methods in detecting zero-day IoT attacks.

Keywords: backdoor; convolutional neural networks; distributed denial of service; machine learning; IDS; IoT; scanning attack; zero-day attacks

check for updates

Citation: Ibrahim Hairab, B.; Aslan, H.K.; Elsayed, M.S.; Jurcut, A.D.; Azer, M.A. Anomaly Detection of Zero-Day Attacks Based on CNN and Regularization Techniques. *Electronics* **2023**, *12*, 573. https://doi.org/10.3390/electronics12030573

Academic Editors: Ayush Goyal and Andrei Kelarev

Received: 5 December 2022
Revised: 10 January 2023
Accepted: 20 January 2023
Published: 23 January 2023

1. Introduction

As Internet of things (IoT) networks start to offer smart and smooth communications to communities and industries, it has become necessary to develop the current lifestyle of the world. With this development, many security concerns have been raised with regard to the privacy protection of users during data transfers and exchanges between the components of those networks.

IoT networks are composed of heterogeneous sensors, processors, and gateways. This introduces multiple types of supported communications due to the different operating systems (OS) that exist in each component/chip. Although IoT networks can be considered a powerful form of communication (transferring data between multiple types of devices), they also have vulnerability issues and limitations with regard to the security types that can be established between both nodes in the network. The security vulnerabilities [1] can lead to many attacks, exposing the users' data to breaches by unknown entities.

The most common attacks in the IoT are denial of service (DoS) and distributed denial of service (DDoS) attacks [2]. These attacks go after the vulnerable nodes in the IoT network and use them later to attack the rest of the network to sabotage specific services and cause data breaches. Another severe attack in IoT is the scanning attack. A scanning attack is used to scan vulnerabilities in a node to decide the best way to attack it later. A backdoor attack establishes unauthorized hidden access mechanisms to keep the target data reachable after the intrusion is done.

A fundamental tool to secure IoT networks is called IDS, which is a software that detects any abnormal activity in the targeted network, IoT or otherwise [3]. IDS can be classified into two classes according to the detection type. The first type is called a signature-based detection, which depends on normal profile activity and compares the current network activity profile; it decides it is an attack if the profiles do not match. This type is considered to be statistical detection. The second type is anomaly-based detection. This type depends on the learning scheme. Because software programs that are trained on available attack data use ML/DL based methods, this type of detection has been used widely in recent years due to its ability to learn and predict the actual data type with high accuracy.

Classical ML-based methods in detection attacks can perform reasonably well with regard to only known data that they have been trained on before. However, these methods cannot distinguish very well between types of data that they have not been trained on yet [4]. On the other hand, DL-based methods have better performance in learning data that give it a better ability to detect unknown data [5], because it is capable of extracting knowledge of nonlinear data to use later to detect unknown data [6]. Convolutional neural networks (CNNs), which are a subcategory of DL, deal with data as images and pixels. They introduce a new feature called parameter sharing, which optimises the number of used parameters to decrease the complexity of the model [7].

Although CNNs belong to the DL category, they still need tuning to be able to differentiate between small variant data of different classes. This is the reason behind using regularization techniques in convolutional neural network (CNN) classifiers, to solve the mentioned issue about overfitting [8]. Consequently, the regularization techniques can help the CNN classifier to better perform in the classification of unknown data by creating a more generic classifier that can be fit on wide range of data classes. Moreover, another approach to avoid overfitting is to add dropout layers in the classifier [9]. Dropout layers work by ignoring weights for random neurons in the classifiers by using a given probability, i.e.: if the probability is given to be 0.25, then 1 weight of each 4 neurons will be set to zero.

The role of regularization techniques in CNN models comes in feature weight configurations to optimize each feature's weight to its ideal value according to its importance. This importance gets continuously measured and updated through the learning/training phase of the model.

In our study, we evaluate the use of CNN models regularized by L1 and L2 methods, respectively, in the TON-IoT dataset to determine their ability in detecting zero-day attacks. The zero-day attacks in our evaluation are backdoor and scanning attacks.

The TON-IoT dataset (2021) [10–15] is a modern IoT dataset containing multiple types of IoT traffic. It simulates the realistic IoT communications by linking three layers: fog, edge, and cloud layer. It also has a variety of attack data available, which make it a good candidate to use in future work extensions and evaluations.

The contributions of this paper are as follows.

1. We evaluate CNN models regularized by L1 and L2 techniques in detecting backdoor and scanning attacks as zero-day attacks by using the network class of the TON-IoT dataset without using any botnet activities as training data. Then, we compare their metrics and results to the nonregularized CNN model and other classical ML models.
2. We prove that the regularized CNN models have a better performance against the standard CNN classifier, due to their higher capability of avoiding overfitting and feature optimization.

3. We prove that regardless of the use of regularization techniques, DL-based methods (CNN classifiers in our case) are better able to decrease the overfitting issue impact than classical ML-based methods.
4. We evaluate whether fixing the overfitting issue in a classifier increases its ability to detect zero-day attacks.
5. We decrease the complexity of security ML-based classifier regularization techniques. This decrease makes it more suitable to IoT deployment because the IoT has less computational power and memory available.

This paper is organized as follows. Section 2 surveys literature regarding other approaches used to detect attacks in the IoT. Section 3 illustrates choosing the used dataset and features alongside the steps for preparing the data in the experiment. Section 4 evaluates, discusses, and demonstrates the final results of the experimental work. Then, conclusions and future work are given in Section 5.

2. Related Work

The detection of IoT network attacks gained broad attention in recent decades as a way to develop better approaches to the protection of users' privacy and data from unknown attackers. In this section, we list recent multiple approaches used to detect IoT attacks in the literature.

2.1. Use of Analytical Approaches

In order to spot IoT botnet attacks, the authors of [16] provided a statistical and analytical method. The intrusion-detection system (IDS) proposed in [17] is a hybrid system based on signatures and anomalies to prevent attacks. Everything up to this point is contained inside the fog layer [18]. This fog layer acts as a bridge between IoT gadgets and the cloud, taking over the duties of detection and monitoring from the gadgets themselves. As a result, they were able to make better use of the hardware. The study used a number of ML algorithms for binary and multiclassification tasks by using the Bot-IoT dataset [19]. Innumerable separate checks were done. XGBoost and DT have superior evaluation metrics than other algorithms, and signature-based detection is faster than anomaly-based detection. Finally, binary classification is preferable than multiclassification for algorithms. Another strategy is using a security information and event-management system to look for botnet activity. A detection system that makes use of IDS, firewalls, operating system networking logs, etc. is proposed in [20]. Input triggers analysis and processing of log files, which the system then uses to establish connections between occurrences. Log files are encrypted and stored depending on the number of packets sent. The network administrator will need to make changes to the firewall to prevent more packets from entering the network if suspicious behavior is discovered. Lightweight and quick identification in IoT networks is possible with the use of statistics. Because these methods are not specifically designed to recognize patterns in data, they may provide subpar forecasts in certain situations.

2.2. Anomaly Approaches

Identifying SYN flood DDoS attacks by using long short-term memory (LSTM) was contrasted in [21] to using a random neural network (RNN) trained simply on regular traffic data. RNNs excel above LSTM in performance. However, RNNs' 81% accuracy is not considered state of the art, despite the fact that it produces better results than LSTM. A unique SVM and CNN-based detection classification approach was proposed in [22]. A 64-by-64-pixel grayscale image is generated from the binary data by the system. These photos are used by CNN and SVM to check whether a file contains malware. For binary classification, the approach has a 94% success rate, whereas multiclassification only has an 81% success rate. The botnet's attacker may only change the file structure, leaving the malware just as dangerous. In other words, this would make system detection better. In [23], the efficient ML technique named Edge2Guard was introduced. The N-BaIoT dataset, which includes both benign and malicious network activity records from the Mirai and Bashlitte botnets, was used for training and testing. PCA reduced 115 characteristics to two. For each MCU-based IoT device, the algorithm generates a unique E2G model to save resources. RF and DT had near-perfect scores. This algorithm's downside is that the model must be improved upon often after being educated by using malware activity data. The authors' suggestion to utilize OTA complicates deployment. In [24], the authors implemented an ML-based forensic approach to identify botnet malware in an IoT network. The analysis shows that although current procedures are effective, they also produce many false positives. The suggested approach captures network packets by using tcpdump. Then, the Bro and Argus tools extract a feature set from the gathered traffic. To classify the data, four basic ML techniques are applied. Methods such as association rule mining (ARM) [25], artificial neural networks (ANNs) [26], naive Bayes (NB) [27], and decision trees (DTs) [28] are used. The Weka tool [29] is then used to evaluate the performance of the intended classifiers in terms of the accuracy, overall success rate (OSR), and false alarm rate (FAR). The decision tree (DT) classifier achieved 93.23% accuracy and 6.77% FAR, according to the authors. ARM exhibited roughly double the accuracy of FAR (13.55% and 86.45%, which are not the greatest DT outcomes).

The particle deep framework (PDF) was proposed in [30]. It had a 99.9% accuracy and 0% FAR. The approach uses deep learning by using a deep multilayer perceptron (MLP) model and particle swarm optimization (PSO) [31] to detect hyperparameters that may boost AUC while training the model. Lastly, the model was trained by using these parameters, which enhanced model training time. The authors in [32] presented a distributed ANN for autoencoder attacks. Autoencoders are a sort of neural network that encodes a distinct input layer for later usage by using sigmoid activation. The suggested autoencoder classifier is operated on each IoT sensor and delivers the cloud with all the records it collects as training data at a greater interval than sensing activity. A CNN-based DL model to identify IoT botnet assaults was proposed in [33]. This model has eight CNN layers that analyze network device power usage. Camera, router, and voice assist devices are IoT sensors on the network, and the botnet was Mirai. The authors' model evaluations demonstrated 96% accuracy. ML-based techniques have a superior capacity to learn from data for future classification decisions [34]. Previous research differs in how training data was chosen, for example, power consumption, network traffic, or the conversion of traffic to grayscale pictures. Deploying ML-based algorithms in IoT networks is difficult. They need a lot of gadget resources, which is not necessarily the case with IoT. This project focuses on constructing an ML-based classification system that avoids overfitting, detects zero-day assaults, and learns nonlinear data relations. A PDF was suggested in [30]. It was 99.9% accurate with no false alarm rate. PSO [31] was used in conjunction with deep learning to identify hyperparameters that may improve AUC during model training, and a deep MLP model was used for the detection of features. Finally, these values were used to train the model. As a result, training models took less time. To counter autoencoder assaults, the authors in [32] introduced a decentralized ANN. One type of neural network, called an autoencoder, uses sigmoid activation to permanently store information about a single input layer. Each IoT sensor runs the proposed autoencoder classifier and uploads its training data to the cloud at a more frequent frequency than detecting activity.

Mbona et al. [35] suggests using semisupervised ML methods to detect zero-day attacks on multiple available datasets. They also utilize Benford's law method [36] to determine the most important features in each dataset to be used in order to optimize the model complexity and performance. The results of the used classifiers against chosen datasets shows limited performance according to the evaluation metrics. It is worth mentioning that the classifiers had better metrics when using the features selected by the Benford's law method.

Hairab et al. [37] conducted a detection scheme by using regularized CNN models to detect DDoS, service scanning and OS fingerprinting as zero-day attacks, although the classifiers were trained on only DoS attack data. The regularized CNN model with the L2 technique outperformed all other classifiers, especially the CNN L2 classifier. Testing the

CNN L2 on OS fingerprinting and DDoS gave accuracy values close to 100%. The data were used from the Bot-IoT dataset [19], which is concerned with attacks launched by botnets in IoT.

For future categorization judgments, ML-based approaches have a greater potential to learn from data [34]. Researchers in the past have used a variety of methods to choose suitable training data. Consider the power drain, the volume of data sent, or the need for grayscale images. It is challenging to implement ML-based algorithms in IoT networks. They demand a lot of hardware, which is not always the case with the IoT. Building an ML-based classification system that can identify zero-day attacks and understand nonlinear data relations is the main goal of this work.

Table 1 shows a comparison between the reviewed approaches from the detection technique and used dataset perspective.

A state-of-art comparison between all the reviewed literature work is presented in Table 2 introducing the advantages and limitations of each.

Table 1. The used datasets and detection techniques of the reviewed approaches.

Reference	Detection Technique					Dataset Used				
	For Zero-Day Attacks	**Supervised**	**Unsupervised**	**Analytical**	**Study Generated Data**	**UNSW-NB15 [38]**	**IoTPOT [39]**	**N-BaIoT [40]**	**Bot-IoT**	**TON-IoT**
Evmorfos et al. [21]		✓			✓					
Su et al. [22]			✓				✓			
Sudharsn et al. [23]		✓						✓		
Al-Duwairi et al. [20]				✓						
Koroniotis et al. [24]		✓	✓			✓				
Koroniotis et al. [30]										
Lawal et al. [16]			✓						✓	
Luo et al. [32]			✓		✓					
Jung et al. [33]			✓		✓					
Mbona et al. [35]	✓	Semi				✓				
Hairab et al. [37]	✓		✓						✓	
Our Evaluation	✓		✓							✓

Table 2. State-of-the-art comparison between the reviewed approaches in detecting IoT attacks and our evaluation.

Reference	Advantages	Limitations
Evmorfos et al. [21]	RNN performed better than LSTM	FPR nearly 19.3% Tested only on one attack.
Su et al. [22]	The developed model is lightweight.	Best accuracy was 94%. Classifier depends on modifiable factors of data.
Sudharsan et al. [23]	Lightweight model for microcontrollers. Each device has a custom trained model.	Deployment requires more efforts as models are different among devices.
Al-Duwairi et al. [20]	IDS processes multiple types of data.	Signature/analytical-based detection that has low learning capabilities.
Koroniotis et al. [24]	Generalized detection to all botnet activities.	The best model (DT) has FAR = 6.77%.
Koroniotis et al. [30]	PSO developed and optimized the accuracy.	Increased the training time for the model after being optimized with PSO.
Lawal et al. [16]	Fog computing is used in the network architecture.	The detection performed less in multiclassification mode.
Luo et al. [32]	Low resource consumption on the targeted devices.	Significant negative impact of noisy data.
Jung et al. [33]	Detection based on power consumption. Model is passive from network perspective, as it does not need to inspect packets.	Accuracy does not exceed 96%.
Mbona et al. [35]	Utilize Benford's law to optimize features.	Limited performance and metrics.
Hairab et al. [37]	High metrics for zero-day attacks for botnet attacks.	Used SMOTE to avoid shortage in normal data. Tested on botnet data only.
Our evaluation	High optimization of used features. High metrics in detecting zero-day attacks. Used on additional zero-day attacks. Tested on nonbot data.	Limited amount of data for specific attacks.

3. Experimental Methodology

For the required evaluation, we use six classifiers to train and test on different data distributions sliced from the TON-IoT dataset. The used classifiers are logistic regression (LR) [41], NB [42], AdaBoost [43], regular CNN, CNN L1, and CNN L2. As the study is directed to zero-day attacks detection, we use for the training phase data that has DDoS attack traffic. The testing phase includes two different data distributions, each in a separate scenario as follows.

1. Scenario A: The used data distribution has backdoor attack data.
2. Scenario B: The used data distribution has scanning attack data.

3.1. The Used CNN Regularization Techniques

L1 technique works best with features selection tasks, as it keeps modifying the weights for the used features until it can be zero, i.e.: neglecting the low important features. The loss function of L1 can be defined as follows:

$$LossFunction = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y} - Y)^2 + \lambda\sum_{i=1}^{N}|\Theta_i|. \tag{1}$$

L2 depends on calculating the square mean of features' weights so it cannot reach zero. Consequently, a feature coefficient can be minimized, but not neglected, as done in L1. We have

$$LossFunction = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y} - Y)^2 + \lambda\sum_{i=1}^{N}\Theta_i^2. \tag{2}$$

We can see that both techniques work on optimizing the weights of the selected features, but when comparing the two, we see that L1 shall perform better when the dataset contains irrelevant features, so they can be totally neglected, whereas L2 would keep a low weight for those features which would decrease the efficiency of the classifier. L2 performs better if all selected features are relevant but with diverged importance.

3.2. Dataset Description

It is important to prove the efficiency of any anomaly-based IDS. This is usually done by using data taken from an available dataset to mock the real scenario in training and testing phases of the classifiers. Thus, choosing the dataset shall be based on its criteria against the required specifications in the study. In our case, we need a modern dataset that has IoT real network traffic data, multiple-attack data available, and a reasonable amount of normal traffic data. We think that the TON-IoT has the required specifications we need to evaluate our classifiers.

- It is a modern dataset generated and published in 2021.
- It has all the attacks data needed in reasonable amounts.
- It has a ground-truth variant of the dataset that tracks all the security intrusion events occurred during the capturing time.
- It has industrial IoT data, network data, Windows data, and Linux data, which make it a good candidate for any future work extensions on the same dataset.
- It contains around 21 million records with 46 features that describe each record.

3.3. Feature Selection

The features of the TON-IoT are organized into seven groups, according to the described activity: connection activity with 12 features, statistical activity with four features, DNS activity with eight features, SSL activity with six features, HTTP activity with 10 features, violation activity with three features, and data labelling group with three features. In our work, we use nine input features against one output feature. To make the classifiers as generic as possible, the features are taken from connection and statistical groups. One of the taken features is `proto` which is a categorical feature that contains values (tcp, udp, and http). We encode this feature into two columns introducing two new columns: `proto_tcp` and `proto_udp` instead of the original proto feature. Table 3 shows the final used features.

Table 3. The chosen set of features from the original features.

Feature Name	**Feature Description**
`duration`	Duration of the flow
`src_bytes`	Bytes from source's sent payload
`dst_bytes`	Bytes responded from receiver's payload
`proto`	The used protocol in this record.
`src_pkts`	Number of packets transmitted from source to destination
`dst_pkts`	Number of packets transmitted from destination
`src_ip_bytes`	Number of bytes of IP header from source.
`dst_ip_bytes`	Number of bytes of IP header from destination.

3.4. Data Preparation

The TON-IoT provides their dataset in a reasonable preprocessed form. The remaining actions to perform are cleaning up the data to make it ready to be processed in the training and testing phases. This clean-up and preparation includes:

- cleaning the dataset from values that do not belong to the accepted ranges of values (noise records);
- picking the chosen features;
- converting nonnumerical values into numerical values;
- standardizing values into one range; and
- preparing three CSV files, one of which has DDoS attack data (for training), and the other two files for the attacks of the backdoor and scanning testing scenarios.

We use the network variant of the TON-IoT dataset in this study and filter the CSV files into three files that each has a specific attack and normal data according to a 2:3 normal:attack data ratio.

The `type` feature has a string value type that contains the attack name in case the `label` equals 1; in case it was 0, the `type` would be normal.

Because the feature `proto` is a categorical feature, and it has string values, it needs to be transformed to another form of data. The transformation has to be to a digital form that can be processed by the used classifiers. Because the available values for this column are only three values—icmp, tcp, and udp—the best way to do this is to encode it to three columns that have a logical type, i.e.: `proto_tcp`, `proto_udp`, and `proto_icmp`. In other words, for each row, only one column among the three columns has value 1, and the other two columns must have 0 value.

The columns must have independent information about each record/row of the dataset. Consequently, one of the three `proto` columns must be removed as it has duplicate information from other two columns. We decide to remove `proto_icmp` and keep the other two columns of `proto`.

Table 4 has the final list of features that are used in this evaluation.

Table 4. The final used features after encoding.

Feature Name	Feature Description
`duration`	Duration of the flow
`src_bytes`	Bytes from source's sent payload
`dst_bytes`	Bytes responded from receiver's payload
`proto_tcp`	Flag indicates if the flow is TCP or not.
`proto_udp`	Flag indicates if the flow is UDP or not.
`src_pkts`	Number of packets transmitted from source to destination
`dst_pkts`	Number of packets transmitted from destination
`src_ip_bytes`	Number of bytes of IP header from source.
`dst_ip_bytes`	Number of bytes of IP header from destination.

For the size of training and testing files, we decide to slice our files, for 350 K records as an upper limit for the training file and 110 K records as an upper limit for the testing file. Final sizes can be less than specified because the dataset does not have this amount of records for all attack types. The final used distributions are in Table 5.

Table 5. The final count of records in all files.

Attack File Name	Number of Records
DDoS (Training)	350 K
Backdoor (Testing 1)	21.5 K
Scanning (Testing 2)	110 K

As a brief for the stage of data preparation, Figure 1 shows the steps of this procedure in order to make the data ready for the next training and testing stages.

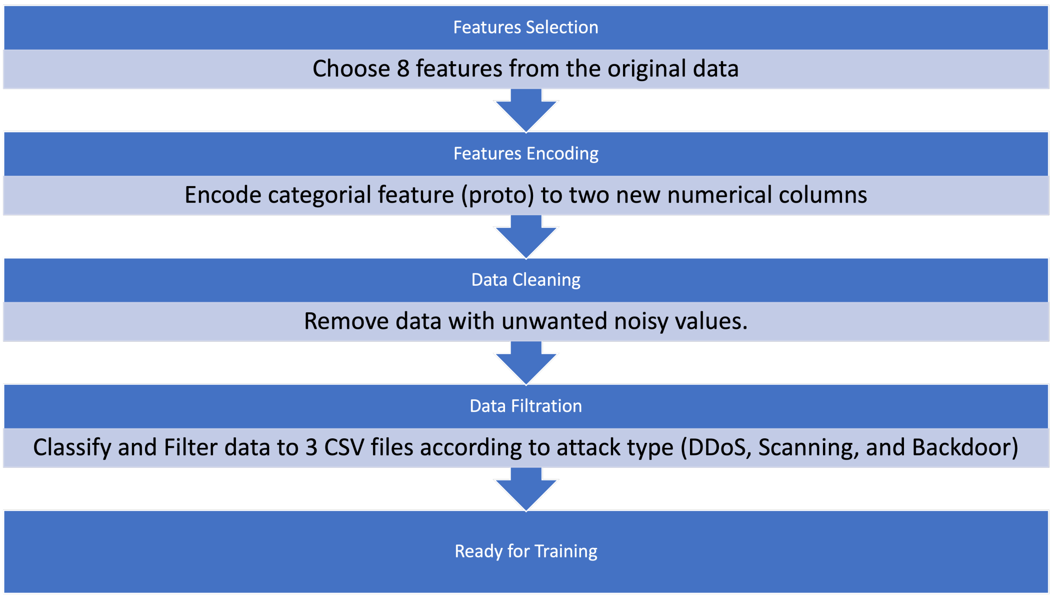
Features Selection
Choose 8 features from the original data
Features Encoding
Encode categorial feature (proto) to two new numerical columns
Data Cleaning
Remove data with unwanted noisy values.
Data Filtration
Classify and Filter data to 3 CSV files according to attack type (DDoS, Scanning, and Backdoor)
Ready for Training

Figure 1. The data preparation process.

3.5. The Architecture of the Used CNN Classifiers

In this section, further details are presented regarding the used three CNN classifiers. Although they have different regularization methods, we use the the same architecture of layers among the three used classifiers. This is done to guarantee measuring the regularization impact on the classifiers' performance. CNN deals with input data as images, so assume our classifiers are dealing with two grayscale images, as we convert each input record to two-dimensional arrays to simulate a grayscale image with one color channel. Each is a 3×3 matrix. To extend the CNN performance evaluation done in [37], in this work we use the same-layers architecture. Our architecture depends on several types of DL and CNN layers. First, the *Shape* layer defines the input of the data that is going to proceed later to the rest of the layers; we define it as $(3 \times 3 \times 1)$. Then, we set two sequenced Conv2D layers. Conv2D is a layer concerned with the number of filters the classifier will learn with regard to the input data. In our case, both Conv2D layers are identical in kernel size of (3×3) and $ReLU$ [44] as an activation function, but with different filters count, which is 32 filters, then 64 filters. Lastly, both layers integrate the required regularization technique depending on the case (no regularization, L1 with factor value of 0.005, or L2 = 0.005). Then, it is conventional to add MaxPooling2D after Conv2D layers to reduce the samples by using a filter that keeps only the maximum value among the filter area which is (2×2) in our case. Then we add a flatten layer to flatten the input shape but with one dropout layer before it to decrease an impact of overfitting. The rate of this dropout layer is 0.3. The rest of the CNN classifiers layers architecture are based on DL, a fully connected dense layer with $ReLU$ activation function, and another dropout layer (rate 0.5). Lastly, the output is presented through a dense layer with units size 1 that is activated through the sigmoid function. A visual representation of this architecture is illustrated in Figure 2. Furthermore, the hyperparameters chosen for these CNN classifiers are presented in Table 6.

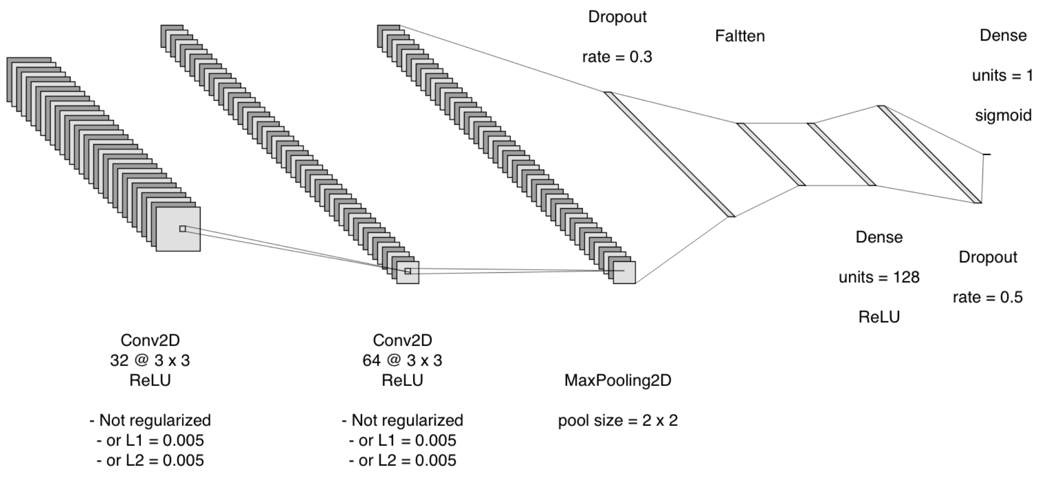

Figure 2. The layers architecture of the three CNN classifiers.

Table 6. The selected hyperparameters to build the CNN classifiers.

Hyper-Parameter	Value
No. Epochs	50
Optimizer	*adam*
Batch	32

3.6. Used Hardware and Software

The described work has been done on a Lenovo PC machine with processor of Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.40 GHz and 8.0 GB RAM operating on Windows 10 Pro 64-bit OS. The detailed machine specifications are illustrated in Table 7. The classifier building process, training, testing, and evaluation are implemented by using Python programming language v3.8 with Spyder IDE integrated in the Anaconda Environment. We also need additional libraries that ease the implementation process which are presented in Table 8.

Table 7. The hardware specifications of the used work station in the experimental work.

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

Table 8. The software libraries used from Python.

Library Name	Version
Keras	2.9.0
Numpy	1.20.3
Pandas	1.3.4
SciKit-Learn	0.24.2
TensorFlow	2.9.0

4. Experimental Work Results

In this section, we conduct our experimental work and illustrate, and the results based on the proper evaluation metrics for each scenario.

4.1. The Evaluation Metrics

Any ML-based classifier, is evaluated based on the confusion matrix [45] that is generated upon any binary classification testing phase. This matrix has four elements that represent the classifier's performance from statistical perspective. Those elements are true positive (TP), false positive (FP), false negative (FN), and true negative (TN).

Those elements help in calculating the final metrics [37] that are used in comparing the classifiers' performances and comparing them to each other. Those metrics are:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

$$F1Score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{6}$$

Receiver Operating Characteristic (ROC): ROC [46] is a curve that plots the true positive rate (TPR) against the false positive rate (FPR), and the area under this curve falls between 0.0 and 1.0, which is an evaluation metric.

4.2. Experimental Results

4.2.1. Scenario A Results

In this scenario, we test the ability of the CNN classifiers detecting the backdoor attack. Table 9 shows reasonable results with high prediction quality of the CNN classifiers compared to the classical ML-based classifiers. Regularized CNN classifiers show even better results than the nonregularized CNN classifier, which has an accuracy of approximately 83%, whereas the regularized classifiers have an accuracy that is 97%. On the other hand, the classical ML classifiers had much lower values of metrics. NB has the best performance among the classical ML-based method, whereas LR and AdaBoost have the least classifiers among all used classifiers. It can be noticed that in general, CNN classifiers have better performance in detecting a backdoor attack as a zero-day attack. Even though classifiers had a reasonable amount of training and testing data in this scenario, the results showed a wide difference between classical ML-based classifiers, the CNN classifier, and the CNN regularized classifiers. This refers to the increasing detection ability in the regularized CNN classifiers.

Table 9. The percentage results of Scenario A.

	Accuracy	Precision		Recall		F1 Score	
		Normal	Attack	Normal	Attack	Normal	Attack
LR	36.55	37.84	1.05	91.28	00.06	53.50	00.11
NB	61.31	87.10	60.84	38.48	99.62	73.71	75.54
AdaBoost	31.29	34.28	0	78.24	0	47.67	0
CNN	83.15	95.89	78.86	60.49	98.27	74.18	87.50
CNN L1	**97.47**	**97.62**	**97.37**	**96.02**	**98.44**	**96.81**	**97.90**
CNN L2	**97.94**	**96.60**	**98.86**	**98.31**	**97.69**	**97.45**	**98.27**

ROC comparison and evaluations which are presented in Figure 3, show the same classifiers order starting from CNN L2 AUC = 0.98, CNN L1 AUC = 0.972; until AdaBoost AUC = 0.391. In general, the higher the AUC, the more that the classifier is able to differentiate between the types and classes of data. The AUC is used to determine the percentage of data that was classified correctly; for example, in our case CNN L2 AUC value means that 98% of data was classified correctly.

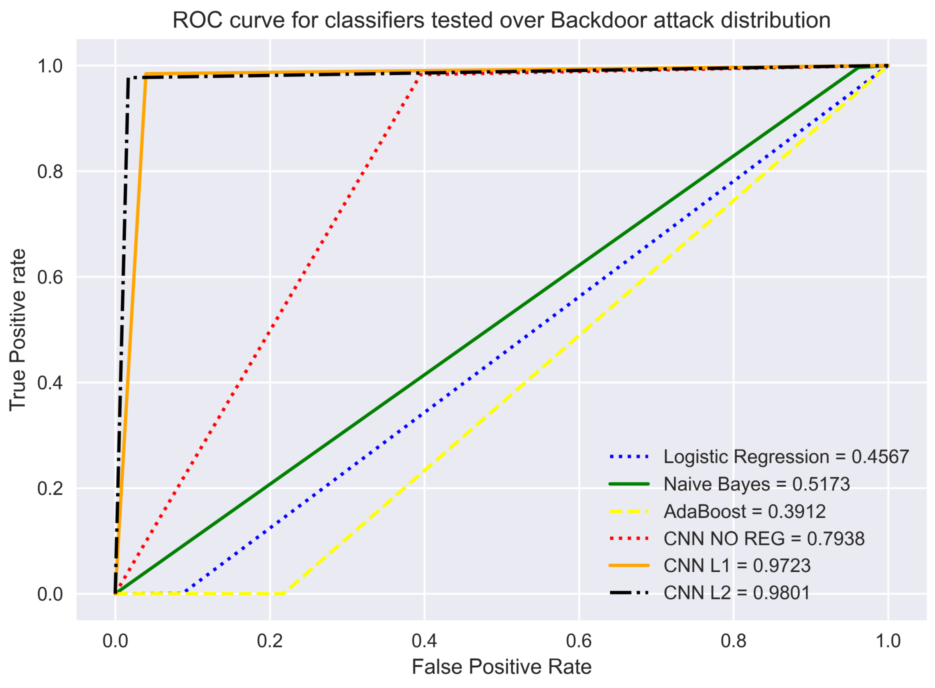

Figure 3. Plot of all classifiers' ROC In Scenario A.

Figure 4 compares between all six classifiers in scenario A regarding AUC, and evaluation metrics.

All in all, with results close enough to 100%, regularized CNN classifiers in this scenario were the best classifiers among the chosen six classifiers regarding all evaluation and comparison aspects.

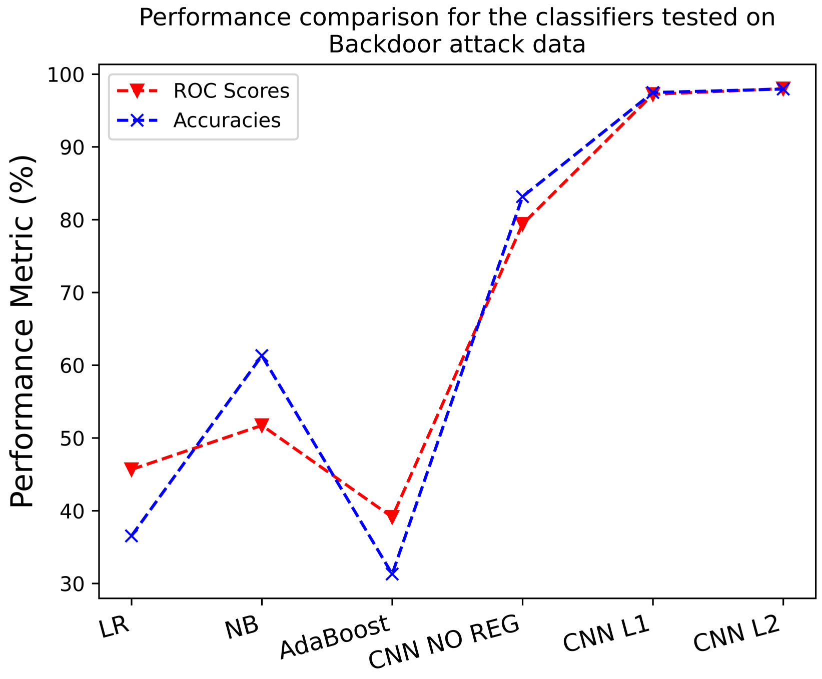

Figure 4. Performance metrics comparison for Scenario A.

4.2.2. Scenario B Results

The second zero attack to be tested in this work is a scanning attack. As in scenario A, in this scenario, regularized CNN classifiers took the lead again in prediction quality as shown in Table 10. The difference is that CNN L1 had a better performance in almost all metrics than CNN L2. The accuracy of CNN classifiers started at 77.52% for nonregularized CNN and reached 87.59% for CNN L1. The classical ML classifiers started at 31.69% for AdaBoost, and reached 61.48% for NB.

Table 10. The percentage results of Scenario B.

	Accuracy	**Precision**		**Recall**		**F1 Score**	
		Normal	**Attack**	**Normal**	**Attack**	**Normal**	**Attack**
LR	37.64	38.27	24.13	91.57	1.78	53.98	3.32
NB	61.48	98.21	60.93	3.63	99.95	7	75.71
AdaBoost	31.69	34.42	4.07	78.44	0.6	47.84	1.05
CNN	77.52	78.29	77.18	60.52	88.83	68.27	82.60
CNN L1	**87.59**	**78.03**	**96.81**	**95.94**	**82.03**	**86.06**	**88.81**
CNN L2	**83.73**	**71.51**	**98.69**	**98.52**	**73.89**	**82.87**	**84.51**

For the second metric, it can be seen in Figure 5 that the classifiers have the same order as the order shown in the metrics Table 10.

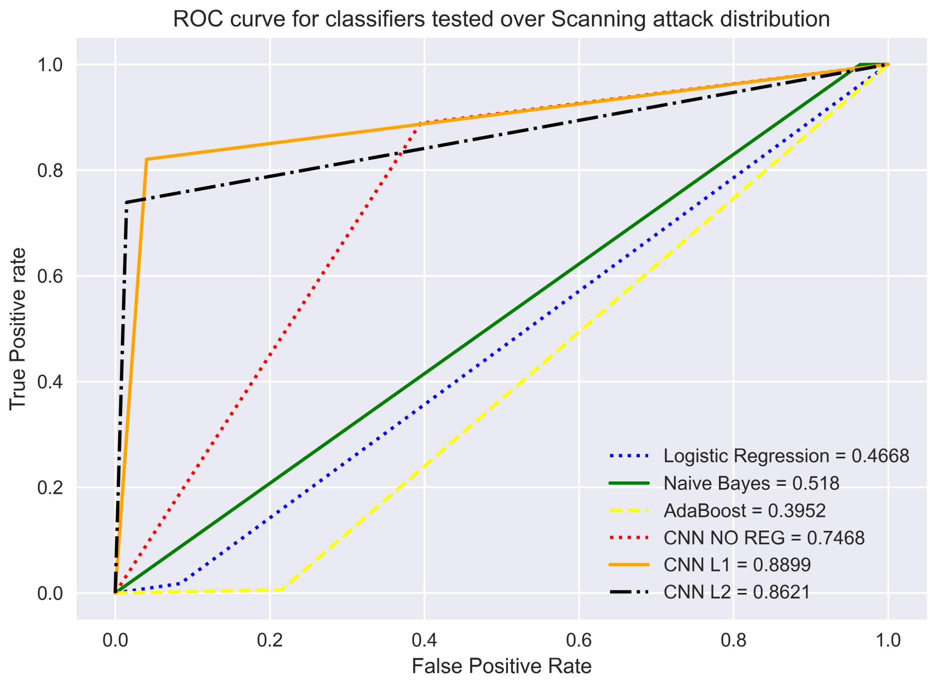

Figure 5. Plot of all classifiers' ROC In Scenario B.

The final full comparison between the classifiers' accuracy values and ROC scores is presented in Figure 6. 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.

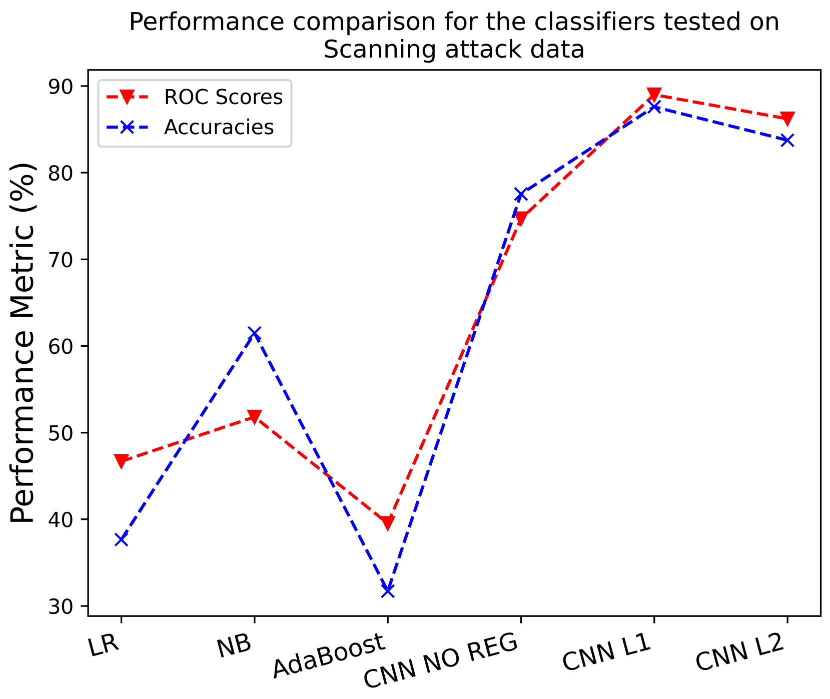

Figure 6. Performance metrics comparison for Scenario B.

The results of the previous scenarios, show that the regularized CNN classifiers perform better than classical ML and DL classifiers in detecting zero-day attacks.

5. Conclusions and Future Work

In this paper, we tested and evaluated regularized CNN classifiers in detecting zero-day attacks by using the TON-IoT dataset. The evaluation included the use of three classical ML-based classifiers and one standard (nonregularized) CNN classifier. The standard CNN classifier is added to measure the gained advantages by using the DL-based method without regularization over classical ML methods, and then the advantages gained by using L1, and L2 regularization techniques in CNN classifiers. In our case, the chosen zero-day attacks were backdoor and scanning attacks. We also tried to highly generalize the classifier learning process by optimizing the dataset features down to only nine input features; most of them are statistical and connectivity features. For training, we trained our classifiers on data containing DDoS attacks and normal data. The final results show a superiority of CNN L2 and CNN L1 when tested on backdoor and scanning attacks. In the backdoor case, CNN L1, and CNN L2 classifiers had an accuracy of approximately 98%. The standard CNN had an accuracy of 83%, whereas other classical ML classifiers' accuracy values ranged between 32% and 61% in their best case. The scanning case also represented a good case for detecting zero-day attacks by using the regularized CNN, where they achieved 83∼87.6% accuracy; and the normal CNN had a 77.52% accuracy. All in all, the mentioned results prove the added advantages of using regularization techniques in CNN classifiers in order to detect zero-day attacks in IoT networks, with respect to the classical ML-based method performance comparison. As a future work, we plan to design, test, and add more tuning to the regularized CNN classifiers to adapt the detection of more zero-day attacks to recent and modern datasets.

Author Contributions: Methodology, B.I.H. and M.S.E.; formal analysis, B.I.H.; writing—original draft, B.I.H.; writing—review and editing, M.A.A.; visualization, A.D.J.; supervision, H.K.A., M.S.E., A.D.J. and M.A.A.; project administration, H.K.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Alqarawi, G.; Alkhalifah, B.; Alharbi, N.; El Khediri, S. Internet-of-Things Security and Vulnerabilities: Case Study. *J. Appl. Secur. Res.* **2022**, 1–17. Available online: https://www.tandfonline.com/doi/citedby/10.1080/19361610.2022.2031841?scroll=top&needAccess=true&role=tab (accessed on 1 December 2022). [CrossRef]
2. Elsayed, M.S.; Le-Khac, N.A.; Dev, S.; Jurcut, A.D. Ddosnet: A deep-learning model for detecting network attacks. In Proceedings of the 2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), Cork, Ireland, 15–18 June 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 391–396.
3. Hairab, B.I.; Azer, M.A. A Distributed Intrusion Detection System for Ad Hoc Networks. In Proceedings of the 2021 16th International Conference on Computer Engineering and Systems (ICCES), Cairo, Egypt, 15–16 December 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–4.
4. El Sayed, M.S.; Le-Khac, N.A.; Azer, M.A.; Jurcut, A.D. A Flow Based Anomaly Detection Approach with Feature Selection Method Against DDoS Attacks in SDNs. *IEEE Trans. Cogn. Commun. Netw.* **2022**, *8*, 1862–1880. [CrossRef]
5. Elsayed, M.S.; Le-Khac, N.A.; Dev, S.; Jurcut, A.D. Detecting abnormal traffic in large-scale networks. In Proceedings of the 2020 International Symposium on Networks, Computers and Communications (ISNCC), Montreal, QC, Canada, 20–22 October 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–7.
6. Said Elsayed, M.; Le-Khac, N.A.; Dev, S.; Jurcut, A.D. Network anomaly detection using LSTM based autoencoder. In Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks, Alicante, Spain, 16–20 November 2020; pp. 37–45.
7. Hwang, R.H.; Peng, M.C.; Huang, C.W.; Lin, P.C.; Nguyen, V.L. An unsupervised deep learning model for early network traffic anomaly detection. *IEEE Access* **2020**, *8*, 30387–30399. [CrossRef]
8. Xiao, Y.; Xing, C.; Zhang, T.; Zhao, Z. An intrusion detection model based on feature reduction and convolutional neural networks. *IEEE Access* **2019**, *7*, 42210–42219. [CrossRef]
9. Li, Y.; Ma, W.; Chen, C.; Zhang, M.; Liu, Y.; Ma, S.; Yang, Y. A Survey on Dropout Methods and Experimental Verification in Recommendation. *arXiv* **2022**, arXiv:2204.02027.
10. Moustafa, N. A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. *Sustain. Cities Soc.* **2021**, *72*, 102994. [CrossRef]

11. Booij, T.M.; Chiscop, I.; Meeuwissen, E.; Moustafa, N.; den Hartog, F.T. ToN_IoT: The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion data sets. *IEEE Internet Things J.* **2021**, *9*, 485–496. [CrossRef]
12. Alsaedi, A.; Moustafa, N.; Tari, Z.; Mahmood, A.; Anwar, A. TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems. *IEEE Access* **2020**, *8*, 165130–165150. [CrossRef]
13. Moustafa, N.; Keshky, M.; Debiez, E.; Janicke, H. Federated TON_IoT Windows datasets for evaluating AI-based security applications. In Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Guangzhou, China, 10–13 November 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 848–855.
14. Moustafa, N.; Ahmed, M.; Ahmed, S. Data analytics-enabled intrusion detection: Evaluations of ToN_IoT linux datasets. In Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Guangzhou, China, 10–13 November 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 727–735.
15. Moustafa, N. New generations of internet of things datasets for cybersecurity applications based machine learning: TON_IoT datasets. In Proceedings of the eResearch Australasia Conference, Brisbane, Australia, 21–25 October 2019; pp. 21–25.
16. Lawal, M.A.; Shaikh, R.A.; Hassan, S.R. An anomaly mitigation framework for iot using fog computing. *Electronics* **2020**, *9*, 1565. [CrossRef]
17. Bridges, R.A.; Glass-Vanderlan, T.R.; Iannacone, M.D.; Vincent, M.S.; Chen, Q. A survey of intrusion detection systems leveraging host data. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 1–35. [CrossRef]
18. Naha, R.K.; Garg, S.; Georgakopoulos, D.; Jayaraman, P.P.; Gao, L.; Xiang, Y.; Ranjan, R. Fog computing: Survey of trends, architectures, requirements, and research directions. *IEEE Access* **2018**, *6*, 47980–48009. [CrossRef]
19. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Gener. Comput. Syst.* **2019**, *100*, 779–796. [CrossRef]
20. Al-Duwairi, B.; Al-Kahla, W.; AlRefai, M.A.; Abdelqader, Y.; Rawash, A.; Fahmawi, R. SIEM-based detection and mitigation of IoT-botnet DDoS attacks. *Int. J. Electr. Comput. Eng.* **2020**, *10*, 2182. [CrossRef]
21. Evmorfos, S.; Vlachodimitropoulos, G.; Bakalos, N.; Gelenbe, E. Neural network architectures for the detection of SYN flood attacks in IoT systems. In Proceedings of the 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments, Corfu, Greece, 30 June–3 July 2020; pp. 1–4.
22. Su, J.; Vasconcellos, D.V.; Prasad, S.; Sgandurra, D.; Feng, Y.; Sakurai, K. Lightweight classification of IoT malware based on image recognition. In Proceedings of the 2018 IEEE 42Nd annual computer software and applications conference (COMPSAC), Tokyo, Japan, 23–27 July 2018; IEEE: Piscataway, NJ, USA, 2018; Volume 2, pp. 664–669.
23. Sudharsan, B.; Sundaram, D.; Patel, P.; Breslin, J.G.; Ali, M.I. Edge2guard: Botnet attacks detecting offline models for resource-constrained iot devices. In Proceedings of the 2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), Kassel, Germany, 22–26 March 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 680–685.
24. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Slay, J. Towards developing network forensic mechanism for botnet activities in the IoT based on machine learning techniques. In Proceedings of the International Conference on Mobile Networks and Management, Melbourne, Australia, 13–15 December 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 30–44.
25. Telikani, A.; Gandomi, A.H.; Shahbahrami, A. A survey of evolutionary computation for association rule mining. *Inf. Sci.* **2020**, *524*, 318–352. [CrossRef]
26. Fan, F.L.; Xiong, J.; Li, M.; Wang, G. On interpretability of artificial neural networks: A survey. *IEEE Trans. Radiat. Plasma Med. Sci.* **2021**, *5*, 741–760. [CrossRef]
27. Umadevi, S.; Marseline, K.J. A survey on data mining classification algorithms. In Proceedings of the 2017 International Conference on Signal Processing and Communication (ICSPC), Tamil Nadu, India, 28–29 July 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 264–268.
28. Priyanka; Kumar, D. Decision tree classifier: A detailed survey. *Int. J. Inf. Decis. Sci.* **2020**, *12*, 246–269. [CrossRef]
29. Singhal, S.; Jena, M. A study on WEKA tool for data preprocessing, classification and clustering. *Int. J. Innov. Technol. Explor. Eng.* **2013**, *2*, 250–253.
30. Koroniotis, N.; Moustafa, N. Enhancing network forensics with particle swarm and deep learning: The particle deep framework. *arXiv* **2020**, arXiv:2005.00722.
31. Sengupta, S.; Basak, S.; Peters, R.A. Particle Swarm Optimization: A survey of historical and recent developments with hybridization perspectives. *Mach. Learn. Knowl. Extr.* **2018**, *1*, 157–191. [CrossRef]
32. Luo, T.; Nagarajan, S.G. Distributed anomaly detection using autoencoder neural networks in WSN for IoT. In Proceedings of the 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.
33. Jung, W.; Zhao, H.; Sun, M.; Zhou, G. IoT botnet detection via power consumption modeling. *Smart Health* **2020**, *15*, 100103. [CrossRef]
34. Elsayed, M.S.; Le-Khac, N.A.; Jurcut, A.D. InSDN: A novel SDN intrusion dataset. *IEEE Access* **2020**, *8*, 165263–165284. [CrossRef]
35. Mbona, I.; Eloff, J.H. Detecting Zero-Day Intrusion Attacks Using Semi-Supervised Machine Learning Approaches. *IEEE Access* **2022**, *10*, 69822–69838. [CrossRef]
36. Mbona, I.; Eloff, J.H. Feature selection using Benford's law to support detection of malicious social media bots. *Inf. Sci.* **2022**, *582*, 369–381. [CrossRef]

37. Hairab, B.I.; Elsayed, M.S.; Jurcut, A.D.; Azer, M.A. Anomaly Detection Based on CNN and Regularization Techniques Against Zero-Day Attacks in IoT Networks. *IEEE Access* **2022**, *10*, 98427–98440. [CrossRef]
38. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 1–6.
39. Pa, Y.M.P.; Suzuki, S.; Yoshioka, K.; Matsumoto, T.; Kasama, T.; Rossow, C. {IoTPOT}: Analysing the Rise of {IoT} Compromises. In Proceedings of the 9th USENIX Workshop on Offensive Technologies (WOOT 15), Washington, DC, USA, 10–11 August 2015.
40. Meidan, Y.; Bohadana, M.; Mathov, Y.; Mirsky, Y.; Shabtai, A.; Breitenbacher, D.; Elovici, Y. N-baiot—Network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Comput.* **2018**, *17*, 12–22. [CrossRef]
41. Maalouf, M. Logistic regression in data analysis: An overview. *Int. J. Data Anal. Tech. Strateg.* **2011**, *3*, 281–299. [CrossRef]
42. Rish, I. An empirical study of the naive Bayes classifier. In Proceedings of the IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, Seattle, WA, USA, 4–6 August 2001; Volume 3, pp. 41–46.
43. Sagi, O.; Rokach, L. Ensemble learning: A survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *8*, e1249. [CrossRef]
44. Dubey, S.R.; Singh, S.K.; Chaudhuri, B.B. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing* **2022**, *503*, 92–108. [CrossRef]
45. Xu, J.; Zhang, Y.; Miao, D. Three-way confusion matrix for classification: A measure driven view. *Inf. Sci.* **2020**, *507*, 772–794. [CrossRef]
46. Bowers, A.J.; Zhou, X. Receiver operating characteristic (ROC) area under the curve (AUC): A diagnostic measure for evaluating the accuracy of predictors of education outcomes. *J. Educ. Stud. Placed Risk* **2019**, *24*, 20–46. [CrossRef]