Received 14 April 2023, accepted 24 April 2023, date of publication 4 May 2023, date of current version 10 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3273160

RESEARCH ARTICLE

An Intelligent Approach to Improving the Performance of Threat Detection in IoT

NGUYEN TAN CAM AND NGUYEN GIA TRUNG
University of Information Technology, Ho Chi Minh City, Vietnam
Vietnam National University, Ho Chi Minh City, Vietnam

Corresponding author: Nguyen Tan Cam (camnt@uit.edu.vn)

This research was supported by The VNUHCM-University of Information Technology's Scientific Research Support Fund.

ABSTRACT Internet of Things (IoT) systems are beneficial to our daily lives and have become increasingly important. A complete IoT system includes devices, sensors, networks, software, and other essential components necessary for operation and interconnection. However, devices and sensors of this nature often have low resource requirements and multiple security vulnerabilities from manufacturers. Moreover, edge network areas of IoT systems exhibit several security weaknesses. Consequently, unauthorized hijacking of sensors or denial-of-service attacks on edge network areas can have severe consequences for the system's operation. In this study, we propose a model that combines machine learning algorithms and principal component analysis techniques to train and predict Distributed Denial of Service (DDoS) attacks. Principal component analysis techniques were applied to reduce data dimensionality. We used accuracy, precision, recall, and F1-Score as the evaluation metrics. We explain the True Positive, False Positive, True Negative, and False Negative measures as basic parts of the above evaluation metrics. Unlike previous studies, we used the Training Time to evaluate the training time of each model. We employed two datasets, CICIDS 2017 and CSE-CIC-IDS 2018, to evaluate our proposed model. In general, the proposed models exhibited the best performance and improved training time.

INDEX TERMS Machine learning, principal component analysis, Internet of Things, DDoS attack.

I. INTRODUCTION

Over the past decade, Industrial Revolution 4.0 has changed the way of doing business and production as well as the ecology of human life. IoT, Artificial Intelligence (AI), Cloud, and Robotic Process Automation (RPA) are the four core elements required for Industry 4.0 success [1]. Smart Home applications with temperature and light sensors can communicate with the central controller in two ways to help automate tasks, such as controlling lights and electronic devices in indoor areas, thereby improving quality of life [2], [3]. The IoT also appears in areas such as physically challenged people, smart health, agriculture, and natural calamities [4]. There were over 13.8 billion IoT devices in 2021 and is predicted to grow to 30.9 billion devices by 2025 [5].

The associate editor coordinating the review of this manuscript and approving it for publication was Wai-Keung Fung.

An IoT system is a network of devices and systems that collaborate to collect, transmit, process, and analyze data from the physical environment. The primary constituents of a complete IoT system are edge devices, gateways, cloud servers, data-analysis tools, and user interfaces. Edge devices are physical devices equipped with sensors, actuators, and controllers to gather and preprocess data and can perform initial analysis before sending it to the cloud. Gateways provide connectivity and processing, acting as intermediaries between edge devices and the cloud to reduce network traffic and latency. Cloud servers store and process data from edge devices and gateways, providing scalable computing resources and storage capacity for handling large volumes of data. Advanced analytics can also be used to obtain detailed insights from the data. Data analysis tools employ machine learning algorithms, statistical analysis, and visualization techniques to identify patterns, trends, and anomalies in data. User interfaces, such as web or mobile applications, enable

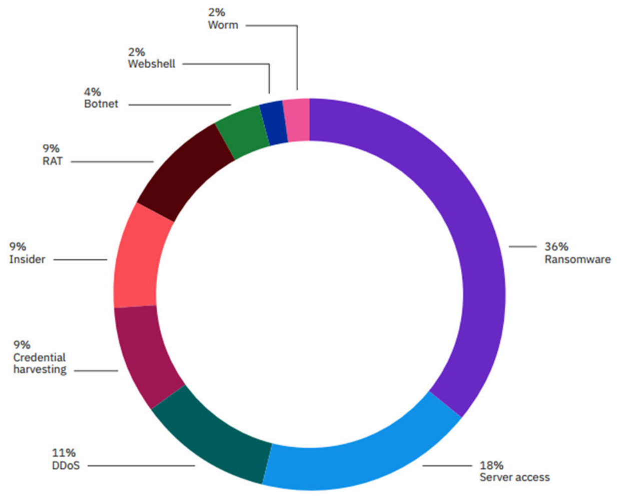

FIGURE 1. Attack types on IoT.

users to interact with the IoT system, access and analyze data from their devices, and receive alerts and notifications based on collected data.

Furthermore, IoT systems are becoming potential targets for cyberattack [6]. From Q3 2019 to Q4 2020, attacks related to IoT systems and the Mozi botnet increased by 3,000% and 74%, respectively. Ransomware, Server Access, and DDoS are common types of attack, as shown in Figure 1 [7]. There are multiple reasons why cyberattackers are increasingly targeting IoT systems, with three main factors being the most important: first, the security vulnerabilities present in sensor devices, often originating from the manufacturer; second, the edge network areas have insufficient protection against cyberattacks; and lastly, the high value of data in these systems makes them attractive targets for attackers.

As a well-established method for bringing down systems since 1973, DDoS continues to be a popular choice for attackers. Since 2017, extensive research has been conducted on DDoS attack methods and techniques in IoT, as well as methods to detect and prevent them [8], [9]. The three main DDoS attack types are volumetric, protocol, and application attacks. These three classifications contain dozens of DDoS attack types and their variants such as UDP, IMCP, IP, TCP, and HTTP floods. Some DDoS attacks fall outside these main categories, and most attackers use a combination of methods to make their attacks more difficult to detect.

The three main approaches for detecting DDoS attacks are anomaly based, signnature-based, and hybrid-based. Signature-based methods rely on specific flow characteristics to build graphs and define a threshold for warning against DDoS attacks [10], [11], [12], [13], [14], [15]. Important features include requests per second, bits per second, and packets per second. Anomaly based methods use machine learning algorithms and training data models to make predictions. However, each method has its own major limitations. The problem of anomaly detection is false positives, whereas that of signature-based detection is a constant change in the attack methods or zero-day DDoS attacks [16]. This poses a challenge to researchers attempting to detect DDoS attacks.

Recently, machine learning and deep learning have been demonstrated to provide accurate predictions for DDoS detection. However, these studies have mostly concentrated on traditional computer networks with powerful servers, which are not applicable to IoT systems that have limited resources on edge devices. In addition, some studies have emphasized only the precision of the algorithm while neglecting the significance of the training time of the model. Therefore, our motivation was to develop a model that could be applied to edge devices. The model must meet two criteria: first, it must have high accuracy in predicting DDoS attacks, and second, it must have fast training time. This solution aims to contribute to building a secure and reliable connection environment for IoT devices, bringing about economic and social benefits, and creating development opportunities for future IoT technologies.

In this article, we are using a combination of Principal Component Analysis (PCA) and Machine Learning (ML) algorithms. The algorithms we used include the Support Vector Machine (SVM), Naïve Bayes (NB), Decision Tree (DT), Random Forest (RF), and Extremely Randomized Trees (ET). First, we evaluated five simple ML by using the original dataset without PCA. We repeated the training and predicted DDoS attacks but with data from PCA. We used the CICIDS 2017 and CSE-CIC-IDS 2018 datasets for evaluation. As a result, our proposed system predicts DDoS attacks with high performance and faster training time than without using PCA.

The remainder of this paper is organized as follows. In the first section, we introduce the current state of cyberattack challenges in IoT environments. Section II analyzes relevant scientific studies. In Section III, we explore the ML models we use. In Section IV, we describe the proposed model. In Section V, we evaluate our model using two datasets, CICIDS 2017 and CSE-CIC-IDS 2018. In Section VI, we present our conclusions and directions for future research.

II. RELATED WORK

In a survey conducted by Wehbi et al. [17], three strategies for detecting DDoS attacks using ML were identified. Each strategy employs a distinct method of integrating machine-learning techniques, including the use of IoT network behavior (Approach 1), software-defined network (SDN) architecture (Approach 2), and Apache Spark (Approach 3). Ziadoon Kamil Masser et al. [18] evaluated ten commonly used supervised and unsupervised ML algorithms for their ability to identify effective and efficient ML-based intrusion detection systems for networks. Using the CICIDS 2017 dataset, this approach yielded the best hyperparameters for each algorithm. Mohammad Najafmehr et al. [19] combined supervised and unsupervised algorithms to detect anomalies in the network traffic. Initially, a Density-Based Spatial Clustering of Applications with Noise (DBSCAN) clustering algorithm was used to separate anomalous traffic from normal data. Then, DT, RF, NB, and SVM machine-learning algorithms were utilized to classify the clusters. The authors evaluated the proposed model on the CICIDS 2017 dataset, and tested it on the more recent CICDDoS 2019 dataset. The results indicate that the Positive Likelihood Ratio of the proposed method is approximately 198% higher than that of traditional ML classification algorithms.

Erhan and Anarim [20] used a hybrid wavelet and matching pursuit algorithm for DDoS detection. The methodology suggested in this study uses a dictionary derived from the network flow parameters. This approach produced over 99% true positives and less than 0.7% false positives. Praseed and Thilagam [21] leveraged a simple annotated Probabilistic Timed Automata (PTA) and a suspicion scoring mechanism to separate normal and anomalous user behavior in the context of asymmetric application-layer DDoS attacks. In their study of Sybil DDoS in the Internet of Vehicles, Li et al. [22] developed a Fast Quartile Deviation Check and an entropy theory-based algorithm to quantify the traffic distribution. The LuST dataset shows that this approach has a detection rate of 100%.

Currently, DDoS attacks with a low-rate bandwidth performance are difficult to detect. Liu et al. [23] used a deep convolutional neural to detect TCP low-rate DDoS attacks from the Mirai botnet. Alsirhani et al. [24] proposed a dynamic solution for detecting DDoS attacks by combining four machine learning algorithms and fuzzy logic. The fuzzy logic system automatically selects the most effective algorithm from a set of predefined classification algorithms to recognize different DDoS attack patterns. Alasmary et al. [25] proposed a ShieldRNN with Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) models. This approach achieved a 99.9% F1-Score. Sayed et al. [26] employed LSTM and Autoencoder (AE) to address the issue of DDoS attacks in SDNs using three datasets: InSDN, CICIDS 2017, and CICIDS 2018. This approach achieved a high detection rate and provided a more efficient method for establishing a model. Beitollah et al. [27] presented an ML solution that merges the Radial Basis Function (RBF) neural network and the Cuckoo Search Algorithm (CSA) to detect App-DDoS traffic. Deepa et al. [28] used embedded learning, combining the K-Nearest Neighbor (KNN), NB, SVM, and Self-Organizing Map (SOM) algorithms to detect DDoS attacks within an SDN controller. The authors evaluated their proposed method on the CAIDA 2016 dataset, and the results showed that the combined learning approach achieved an accuracy of 97-98%, whereas traditional machine learning methods without combination only achieved an accuracy of 75-84%. Soe et al. [29] used an Artificial Neural Network (ANN) algorithm combined with the synthetic minority oversampling technique (SMOTE) to reduce data imbalance in DDoS detection. Jia et al. [30] proposed an LSTM-based FlowGuard model to detect DDoS attacks. The model was evaluated on the dataset CICDDoS 2019 with an accuracy of up to 98.9%. For instance, Khedkar and AroulCanessane [31] used an SVM model to classify attacks, and achieved an accuracy of 78%. Additionally, Zeeshan et al. [32] proposed a protocol-based architecture and detected DDoS attacks using a deep learning LSTM. The classification accuracy was 96.3% for the UNSWNB15 and Bot-IoT datasets.

In their research, Sharafaldin et al. [33] compared the different datasets. The author pointed out four important features for detecting DDoS attacks: Standard Deviation size of Packet, Packet Size, Flow Duration, Standard Deviation time between two packets in flow. The authors used the ML algorithm on these four selected features and obtained a 98% accuracy. Unlike previous works that concentrated mainly on DDoS attacks launched from compromised IoT devices towards servers, Tushir et al. [34] focused on the connectivity and energy consumption of IoT devices during attacks. Key discoveries include buffer overflow and the group key updating process of WiFi.

Deep learning (DL) algorithms have also been widely applied in research related to network intrusion detection systems (NIDS) or host intrusion detection systems (HIDS). Vinayakumar et al. [35] utilized a Deep Neural Network (DNN) algorithm to detect malware attacks on both HIDS and NIDS. They used the KDDCup 99 dataset to identify optimal hyperparameters for the algorithm. They then tested the algorithm with these hyperparameters on various datasets, including NSL-KDD, UNSW-NB15, Kyoto, WSN-DS, and CICIDS 2017, and demonstrated a good predictive performance. Hnamte and Hussain [36] employed a hybrid DCNNBiLSTM model that integrated CNN, LSTM, and DNN. Their methodology was evaluated on the CSE-CIC-IDS 2018 and Edge_IIoT datasets, resulting in accuracy rates of 100% and 99.64%, respectively. Additionally, Okey et al. [37] proposed an embedded model called ELETL-IDS, which was built using model averaging with three preselected models: InceptionV3, MobileNetV3Small, and EfficientNetV2B0. The proposed model was evaluated on two datasets, CICIDS 2017 and CSE-CIC-IDS 2018, and it achieved perfect results for all evaluation metrics, including 100% accuracy, precision, recall, and F-score. Wei et al. [38] combined the AE model and multilayer perceptron network (MLP) to evaluate DDoS attacks on the CICDDoS2019 dataset, obtaining an accuracy of 98%. To detect DDoS attacks in SDN, Haider et al. [39] proposed four DL-based architectures by using three DL algorithms (RNN, LSTM, and CNN): embedded RNN (RNN + RNN), embedded CNN (CNN + CNN), embedded LSTM (LSTM + LSTM), and hybrid embedded (RNN + LSTM). Among these models, the embedded CNN architecture exhibited the highest performance, achieving an accuracy of 99.45% when evaluated on the CICIDS 2017 dataset. Liu et al. [40] tested the LSTM algorithm on an asynchronous federated learning platform utilizing the ISCX-2016-SlowDos and 1999 DARPA datasets to combat low-rate DDoS attacks and achieved a remarkable accuracy rate of 98.8%. In a previous study by Roopak et al. [41] evaluated DDoS attacks on the CICIDS2017 dataset using four DL models: MLP, 1d-CNN, LSTM, and CNN + LSTM. The results showed that the LSTM + CNN model performed the best with an accuracy of 97.16%.

In general, research works in recent years have shown positive results, but there are still some disadvantages; for example, using individual datasets makes the model evaluation not high fidelity, and many studies focus on accuracy, precision, recall, or F1-Score quantities and ignore the training time of algorithms, a factor that is also very important in attack detection in the IoT environment, as detailed in Table 1.

III. BACKGROUND METHODOLOGY

We used five Simple Machine Learning algorithms: Naïve Bayes (NB), Decision Tree (DT), Random Forest (RF), Extremely Randomized Trees (ET), and Support Vector Machine (SVM), in combination with the Principal Component Analysis (PCA) technique for implementing DDoS attack detection.

A. MACHINE LEARNING ALGORITHMS

1) NB

The NB algorithm assumes that the data features are independent of each other. Instead of trying to find the label for each data point, NB looks for the most likely label to be assigned to the data point using the following formula:

$$c = argmax\ p(c|x) \tag{1}$$

where $p(c|x)$ is the probability of label c of data point x. According to Bayes' rule, the above formula can be re-expressed as follows:

$$c = argmax \frac{p(c|x) \times p(c)}{p(x)} \tag{2}$$

where $p(c)$ is calculated according to the method of Maximum Likelihood Estimation (MLE) or Maximum A Posteriori (MAP).

Owing to the assumption that the data features are independent of each other, the algorithm can easily choose label c for the data, making the training and testing times extremely fast.

2) DT

The DT algorithm is a supervised learning algorithm. The unit of DT is the node, which is described as:

$$node = (x_i, t_i, c_i) \tag{3}$$

where x_i is the data value at node i and t_i is the criterion for selecting the next node of node i. If $i = 1$, then it is the root node. If $t_i \neq 0$, then node is a nonleaf node. If $t_i = 0$, then that node is leaf node. At the leaf node, the data are labeled c_i. The choice of the value of the t_i criterion is based on the evaluation index entropy and information gain. The DT architecture is illustrated in Figure 2.

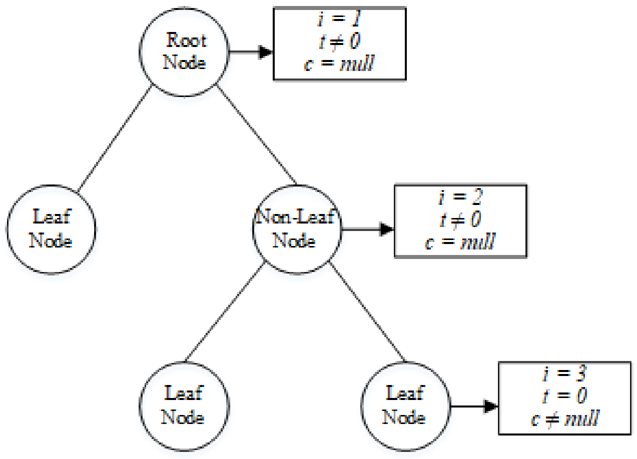

FIGURE 2. DT architecture.

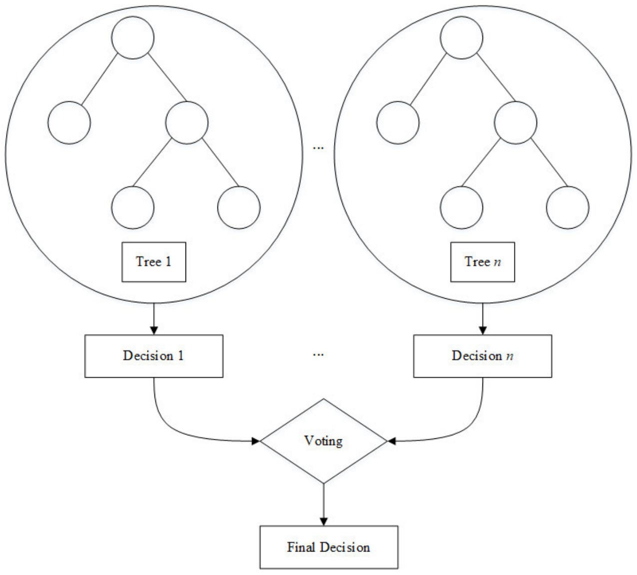

FIGURE 3. RF architecture.

If the DT algorithm is sufficiently deep, the model will easily encounter overfitting.

3) RF

The RF approach involves a multitude of individual DT that together function as an ensemble. Each DT generates a class prediction, and the prediction with the most votes corresponds to that of the model. The wisdom of crowds states that a committee of many uncorrelated models (trees) will outperform a single model. Therefore, RF has full DT properties and better solves the overfitting problem of DT. The RF architecture is shown in Figure 3.

4) ET

Similar to RF, ET comprises many DT and is an ensemble algorithm. Therefore, ET has full DT properties. However, ET uses more random thresholds for each feature instead of

TABLE 1. Comparison of other related techniques.

Article	Dataset	Algorithms	Accuracy	Precision	Recall	F1-Score	Training Time (s)
Ziadoon Kamil Masser *et al.* [18]	CICIDS 2017	ANN	0.9928	0.9937	0.9928	0.9917	53.78
		DT	0.9949	0.9943	0.9949	0.9942	1.23
		k-NN	0.9952	0.9949	0.9952	0.9949	11.13
		NB	0.9886	0.9901	0.9886	0.9885	1.07
		RF	0.9930	0.9909	0.9930	0.9912	9.38
		SVM	0.7521	0.9916	0.7521	0.7660	343.56
		CNN	0.9947	0.9943	0.9946	0.9944	261.8
Mohammad Najafmehr *et al.* [19]	CICIDS 2017 CICDDoS2019	DBSCAN + RF	0.1479	0.9989	0.1454	No	No
		DBSCAN + SVM	0.40100	0.99848	0.39970	No	No
Derya Erhan *et al.* [20]	CAIDA 2016	Wavelet + Matching Pursuit	0.9979	No	No	No	No
Amit Praseed *et al.* [21]	SDSC-HTTP CLARKNET-HTTP	PTA	0.9980	0.9993	No	0.9989	No
Amjad Alsirhani *et al.* [24]	CAIDA 2016	NB	0.622	0.625	0.622	0.544	No
		DT	0.97	0.97	0.97	0.97	No
		RF	0.972	0.973	0.972	0.973	No
Faris Alasmary *et al.* [25]	CICIDS 2017	RNN + LSTM	1	1	1	1	No
Mahmoud Said El Sayed *et al.* [26]	InSDN	LSTM + AE	No	0.9993	0.9993	0.9993	No
	CICIDS 2017		No	0.9979	0.9996	0.9988	No
	CSE-CIC-IDS 2018		No	1	0.9999	0.9999	No
Hakem Beitollah *et al.* [27]	NSL-KDD	RBF + CSA	0.969	0.971	No	No	No
V.Deepa *et al.* [28]	CAIDA 2016	SVM + SOM	0.9812	0.9714	No	No	No
Yan Naung Soe *et al.* [29]	Bot-IoT	SMOTE + ANN	1	1	1	1	No
Yizhen Jia *et al.* [30]	CICDDoS2019	LSTM + CNN	0.989	0.9947	0.9931	0.9935	No
Shilpa P Khedkar *et al.* [31]	Not Available	SVM	0.7832	0.81	0.99	0.89	No
Muhammad Zeeshan *et al.* [32]	UNSWNB15 Bot-IoT	LSTM	0.963	No	No	No	No
Iman Sharafaldin *et al.* [33]	CICIDS 2017	KNN	No	0.96	0.96	0.96	1,908.23
		RF	No	0.98	0.97	0.97	74.39
		NB	No	0.88	0.04	0.04	14.77
Vanlalruata Hnamte *et al.* [36]	CSE-CIC-IDS 2018	CNN + LSTM	1	No	No	No	202
	Edge_IIoT		0.9996	No	No	No	500
Ogobuchi Daniel Okey *et al.* [37]	CICIDS 2017	InceptionV3 + MobileNetV3Small + EfficientNetV2B0	1	1	1	1	36,366
	CSE-CIC-IDS 2018		0.9999	0.9999	0.9999	0.9999	17,695
Yuanyuan Wei *et al.* [38]	CICDDoS2019	AE + MLP	0.9834	0.9791	0.9848	0.9818	No

finding the best threshold, such as RF. ET usually has a faster training time than RF because it saves time when finding thresholds for features at nodes.

5) SVM

The SVM algorithm is based on the concept that data classes are classified and laid out in linearly separable planes. We can now determine an infinite number of planes that divide these data classes. The distance from the nearest data point to the separation plane is the margin. The SVM looks for a hyperplane such that the margin is at its maximum and the margins of the data layers are the same. This hyperplane can be described as follows.

$$w^T x + b = 0 \tag{4}$$

The SVM algorithm used to find the value *(w, b)* is represented by the following formula:

$$(w, b) = argmax\{min(margin)\} \tag{5}$$

SVM algorithms face many challenges with data classes with many outliers.

B. PRINCIPAL COMPONENT ANALYSIS

The PCA algorithm is based on the mathematical idea that data are not normally distributed randomly but are instead distributed according to a coordinate system. Assume that the initial data have a characteristic D with $D \in R$. PCA projects the data onto a new space, where the data are represented with the characteristic K with $K \in R$ and $K \ll D$. K is also known as the principal component.

The value of K is determined based on the level of information that must be preserved. Here, the total information is the sum of variances in all dimensions of the data. The amount of data to be maintained is the total variance of the new coordinate system. Therefore, to choose K, we can rely on the following formula:

$$R_K = \frac{\sum_{i=1}^{K} \lambda_i}{\sum_{i=1}^{D} \lambda_i} \tag{6}$$

where R_K is the amount of data to be kept, and λ is the variance of the data.

IV. PROPOSED SYSTEM

Our study presents the system depicted in Figure 4, which comprises four main steps: raw data, data cleaning, pre-processing, and detection.

A. RAW DATA

Raw data are collected from IoT devices in the form of Events, Flows or Packets. In which:

An event represents a log of a specific occurrence, such as a user login or VPN connection, which is documented at the moment it occurs.

A packet refers to a piece of data traveling from the sender to its destination, which moves from the source address to the destination address via intermediate hops. The combination of multiple packet segments results in a larger number of messages at the receiver address.

Flow: Set of packets that share seven attributes. Once any of these attributes changes, a new flow is established. The seven attributes are incoming traffic interface, source IP address, destination IP address, IP protocol, source port, destination port, and IP service type.

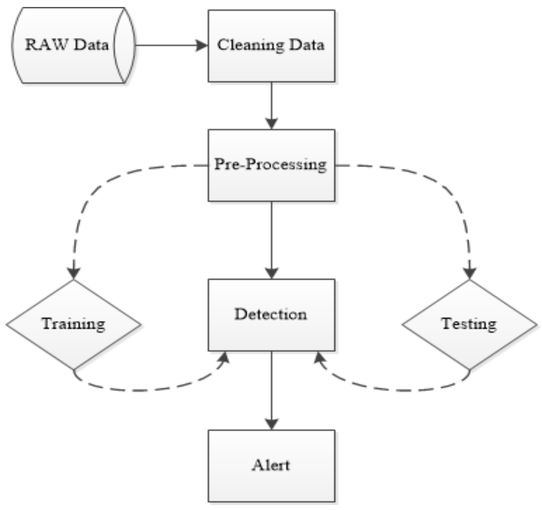

FIGURE 4. Proposed system.

In many DDoS attacks, flow measurements are important to the outcome. However, with the sophistication of attacks, Event or Packet information also contributes to detection.

B. CLEANING DATA

In any machine-learning project, raw data are often unsuitable for direct use for several reasons, including the requirement for numerical data by machine-learning algorithms, presence of statistical noise, errors, missing values, and conflicting examples, as well as the need to uncover complex nonlinear relationships within the compressed raw data. In addition, categorical data must be converted into numerical values to ensure compatibility with machine learning algorithms.

Cleaning the data is a critical step. The success of a machine learning algorithm depends on the quality of the training data. In this study, we used the datasets CICIDS 2017 and CSE-CIC-IDS 2018. With this dataset, we removed the data with little information and perturbed the model to obtain a dataset with high value. The steps are as described in Algorithm 1.

C. PRE-PROCESSING

The data were separated into a training set (70%) and testing set (30%), and model training was conducted only on the training set. The testing set was used solely for testing.

Scaling: Because the input variables of the data were different, we scaled the data to help the algorithms learn better. A normalization technique was used to achieve this. Data normalization involves scaling the data from their original range to a new range, which is typically between 0 and 1.

$$y = \frac{x - min}{max - min} \tag{7}$$

where the maximum and minimum values pertain to the value x being normalized.

PCA: We implemented the PCA dimensionality reduction technique to improve the performance and training time of the model, which is more suitable for detecting DDoS attacks in IoT environments. Incorporating PCA techniques also helps

Algorithm 1 Cleaning Data Algorithm

```
Input: original_dataset: original dataset containing
       features and samples.
       threshold: a threshold for variance below
       which features are removed.
Output: cleaning_dataset: dataset after cleaning.
1  Begin
2  for feature in original_dataset.features:
3     unique_values = count_unique_values(feature)
4     variance = calculate_variance(feature)
5     if unique_values == 1 or variance < threshold:
6        delete_feature(feature)
7    for sample in original_dataset.samples:
8       if duplicate(sample) or
          missing_ infinite(sample):
9          delete_sample(sample)
10      elif sample.label == "Benign":
11         sample.label = 0
12      elif sample.label != "Benign":
13         sample.label = 1
14   Return: cleaning_dataset
15   End
```

where count_unique_values() is the function reporting the number of unique values in each feature; calculate_variance() is the function that calculates the variance in each feature; duplicate() checks whether it is a duplicate; missing_infinite() is the function that checks whether it contains any missing or infinite values; delete_feature() is the function to delete a feature; delete_sample() is the function to delete a sample.

models avoid outliers that are commonly found in machine learning data. We chose K principal components of the data, based on the amount of variance retained. The pre-processing steps are described in Algorithm 2.

Algorithm 2 Pre-Processing Algorithm

```
Input: cleaning_dataset: dataset after cleaning in
       Algorithm 1.
       n_components: the amount of variance retained:
       95%, 90%, 85%, 80%, and 75%, respectively.
Output: X_Train_withoutPCA: training set features do
        not include labels without PCA.
        X_Test_withoutPCA: testing set features do not
        include labels without PCA.
        X_Train_withPCA: training set features do not
        include labels with PCA.
        X_Test_withPCA: testing set features do not
        include labels with PCA.
        Y_Train: training set labels.
        Y_Test: testing set labels.
1  Begin
2  X_Train, X_Test, Y_Train, Y_Test
   = train_test_split(cleaning_dataset, test_size = 0.3)
3  X_Train_Scaler=X_Train.minmaxscaler
   (feature_range =(0,1))
4  X_Test_Scaler=X_Test.minmaxscaler
   (feature_range =(0,1))
5  if not using PCA:
6     X_Train_withoutPCA =X_Train_Scaler
7     X_Test_withoutPCA =X_Test_Scaler
8     Return: X_Train_withoutPCA,
      X_Test_withoutPCA,
      Y_Train, Y_Test
9  if using PCA:
10    for K in n_components:
11       X_Train_withPCA = pca (X_Train_Scaler, K)
12       X_Test_withPCA = pca (X_Test_Scaler, K)
13    Return: X_Train_withPCA, X_Test_withPCA,
      Y_Train, Y_Test
14 End
```

where train_test_split() is the function for splitting the dataset into a training set and a testing set; the minmaxscaler() function is used to transform all feature values to a range between 0 and 1; pca() is the function that reduces dataset dimensionality.

D. DETECTION

After the cleaning and pre-processing steps, the data were used to train the machine learning model. To help identify the proposed model, we first trained and evaluated the machine learning models using data obtained after cleaning and scaling without PCA. Then, we continued to train and evaluate the machine learning models with five cases of principal component K with the amount of variance retained: 95%, 90%, 85%, 80%, and 75%, respectively. Therefore, using five machine-learning models, we have done thirty times of training and evaluation times to determine the best model for our proposal. These steps are presented in Algorithm 3.

V. EVALUATION

A. DATASET

To evaluate our proposed system, we conducted experiments on two datasets, CICIDS 2017 and CSE-CIC-IDS 2018. The CICIDS 2017 dataset was publicly released in 2017 by the Canadian Institute for Cybersecurity (CIC), whereas the Communications Security Establishment (CSE) and CIC jointly released the CSE-CIC-IDS 2018 dataset in 2018. We chose these datasets for three reasons: First, they are current network-based datasets that are appropriate for appraising the detection of DDoS attacks, which is the primary focus of this study [42]. Second, this dataset was confirmed to meet 11 IPS dataset criteria: complete network configuration, complete traffic, labeled data, complete interaction, complete capture, available protocols, attack diversity, anonymity, heterogeneity, feature set, and metadata [43]. Finally, to construct this dataset, the author fully deployed

Algorithm 3 Detection Algorithm

```
Input: X_Train_withoutPCA: training set features do
         not include labels without PCA.
       X_Test_withoutPCA: testing set features do not
         include labels without PCA.
       X_Train_withPCA: training set features do not
         include labels with PCA.
       X_Test_withPCA: testing set features do not
         include labels with PCA.
       Y_Train: training set labels.
       Y_Test: testing set labels.
       Algorithms: machine learning algorithm includes
         NB, DT, RF, ET, SVM.
       n_components: the amount of variance retained:
         95%, 90%, 85%, 80%, and 75%, respectively.
Output: proposed_model.
1  Begin
2  if not using PCA:
3    for algorithm in Algorithms:
4      startTT = time. time()
5      algorithm. fit (X_Train_withoutPCA, Y_Train)
6      endTT = time. time()
7      Y_predict = algorithm.
       predict (X_Test_withoutPCA)
8      accuracy = accuracy_score (Y_predict, Y_Test)
9      precision = precision_score (Y_predict, Y_Test)
10     recall = recall_score (Y_predict, Y_Test)
11     f1 = f1_score (Y_predict, Y_Test)
12     TrainingTime = (endTT – startTT)
13   Return: Algorithms_Performance_withoutPCA(
       accuracy, precision, recall, f1, TrainingTime)
14 if using PCA:
15   for K in n_components:
16     for algorithm in Algorithms:
17       startTT = time. time()
18       algorithm.fit(X_Train_withPCA, Y_Train)
19       endTT = time. time()
20       Y_predict = algorithm.
         predict(X_Test_withPCA)
21       accuracy = accuracy_score (Y_predict, Y_Test)
22       precision = precision_score
         (Y_predict, Y_Test)
23       recall = recall_score (Y_predict, Y_Test)
24       f1 = f1_score (Y_predict, Y_Test)
25       TrainingTime = (endTT – startTT)
26     Return: Algorithms_Performance_withPCA(
         accuracy, precision, recall, f1, TrainingTime)
27   Return: Algorithms_Performance_withPCA_for
       _K( accuracy, precision, recall, f1, Training
       Time)
28 proposed_model =
   evaluate (Algorithms_Performance
   _withoutPCA, Algorithms
   _Performance_withPCA_for_K)
29 Return: proposed_model
```

Algorithm 3 *(Continued.)* Detection Algorithm

```
30 End
```

where time() is the function that gets the current time of the system; fit() is the function that trains the model; predict() is the function that makes predictions using a trained model; accuracy_score() is the function to evaluate the accuracy measure of the trained model; precision_score() is the function to evaluate the precision measure of the trained model; recall_score() is the function to evaluate the recall measure of the trained model; f1_score() is the function to evaluate the harmonic mean of the precision and recall measure of the trained model; evaluate() is the function to evaluate the performance of machine learning algorithms with and without PCA.

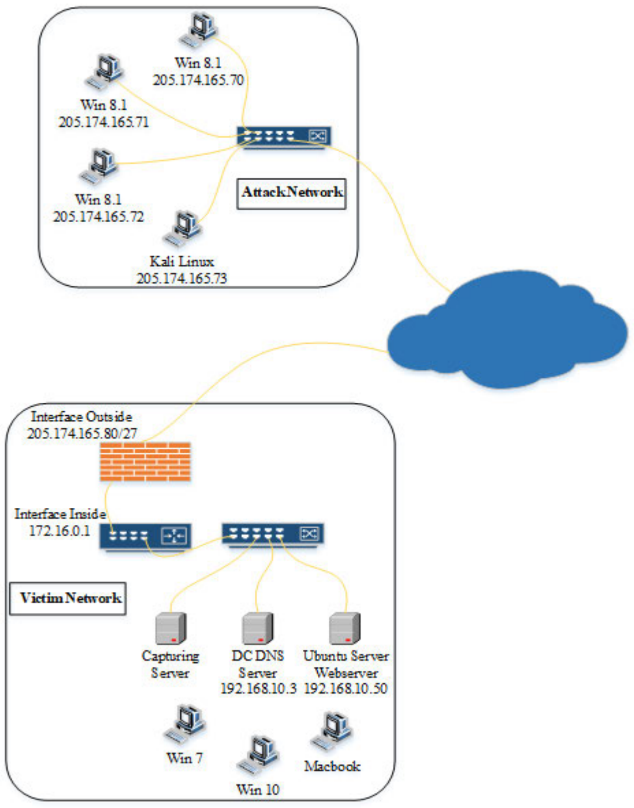

FIGURE 5. CICIDS 2017 topology.

modern attack techniques in the system, including firewalls, routers, switches, and several different operating systems. The victim and attacker zones were completely separated to obtain data similar to real-world scenarios. The model used to build the dataset is shown in Figure 5.

The proposed model was evaluated using the "Friday-WorkingHours-Afternoon-DDoS.pcap_ISCX.csv" file from the CICIDS 2017 dataset and the "02-21-2018.csv" file from the CSE-CIC-IDS 2018 dataset to detect the DDoS attacks. The CICIDS 2017 dataset contains 225,745 samples and 79 features. After applying the cleaning data, the number of features was 44, and the number of samples was 221,125. It included 128,014 DDoS and 93,111 benign records. The CSE-CIC-IDS 2018 dataset consists of 1,046,845 samples and 80 features. Following data cleaning, the dataset was reduced to 559,651 samples with 21 features, comprising 198,861 DDoS records and 360,790 benign records. Note that the number of features above includes the labels. The results are presented in Table 2.

The features after cleaning the CICIDS 2017 and CSE-CIC-IDS 2018 datasets are listed in Table 3.

B. EVALUATION METRICS

We measured the performance of the proposed system using four metrics: true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). In which:

TP: is the measure of the correct prediction of attack samples.

FP: is the measure of the false prediction of attack samples (i.e., benign samples that guess an attack).

TN: is a measure of the correct prediction of benign samples.

FN: is a measure of the false prediction of benign samples (i.e., an attack that guesses them as benign).

We evaluate the Accuracy of the model according to the formula:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{8}$$

We also focused on the precision measure to evaluate the model's ability to accurately predict attacks, as well as recall, to evaluate the omission of attack data:

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

We double check the comprehensiveness of the model with F1-Score:

$$F1_{Score} = 2 \times \frac{(Precision + Recall)}{(Precision * Recall)} \tag{11}$$

Furthermore, we are particularly concerned with evaluating the amount of time each model takes to train using the following formula:

$$TrainingTime(TT) = endTT - -startTT \tag{12}$$

To compare the results of our proposed model and ML algorithms without PCA, we used a quantity called the change magnitude C. The formula used was as follows:

$$C_{Accuracy} = 100 \times \frac{Accuracy_{OurProposed} - Accuracy_{Algorithms}}{Accuracy_{Algorithms}} \tag{13}$$

$$C_{Precision} = 100 \times \frac{Precision_{OurProposed} - Precision_{Algorithms}}{Precision_{Algorithms}} \tag{14}$$

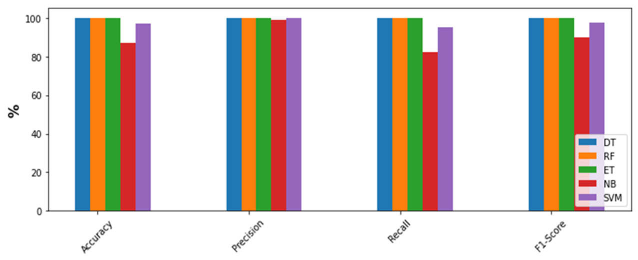

FIGURE 6. Performance comparison of the machine learning model without PCA with CICIDS 2017.

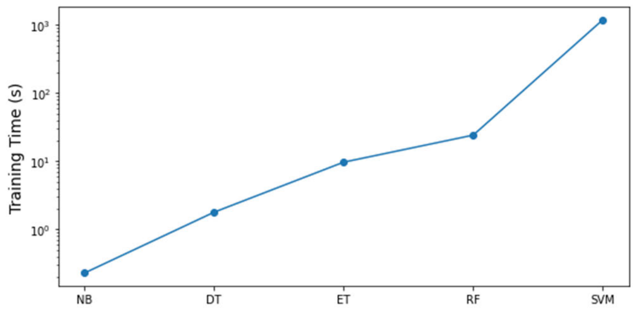

FIGURE 7. Training time comparison of the machine learning model without PCA with CICIDS 2017.

$$C_{Recall} = 100 \times \frac{Recall_{OurProposed} - Recall_{Algorithms}}{Recall_{Algorithms}} \tag{15}$$

$$C_{F1-Score} = 100 \times \frac{F1 - Score_{OurProposed} - F1 - Score_{Algorithms}}{F1 - Score_{Algorithms}} \tag{16}$$

$$C_{TT} = \frac{TrainingTime_{Algorithms}}{TrainingTime_{OurProposed}} \tag{17}$$

C. RESULTS WITHOUT PCA

As a first step in evaluating the proposed system, we performed a DDoS attack detection assessment on the CICIDS 2017 dataset with 44 data features after data cleaning. The DT, RF, and ET algorithms exhibited more than 99.9% performance results, including Accuracy, Precision, Recall, and F1-score. The SVM algorithm had 97% accuracy and the NB algorithm had 87% accuracy. The performance results are shown in Figure 6. The training time of the NB algorithm gives the best results at 0.23 seconds; the DT, ET, and RF are 1.79 seconds, 9.72 seconds, and 24.21 seconds, respectively. The SVM algorithm had a training time of up to 1,190 seconds, as shown in Figure 7. We conducted a similar evaluation on the CSE-CIC-IDS 2018 dataset; the results for both datasets are presented in Table 5.

D. RESULTS WITH PCA

After that, we evaluated the ML algorithms, including DT, RF, ET, NB, and SVM using the PCA technique. The retained

TABLE 2. Comparison of data cleaning.

		Samples			Features
		DDoS	Benign	Total	
CICIDS 2017	Before Cleaning Data	128,027	97,718	225,745	79
	After Cleaning Data	128,014	93,111	221,125	44
CSE-CIC-IDS 2018	Before Cleaning Data	686,012	360,833	1,046,845	80
	After Cleaning Data	198,861	360,790	559,651	21

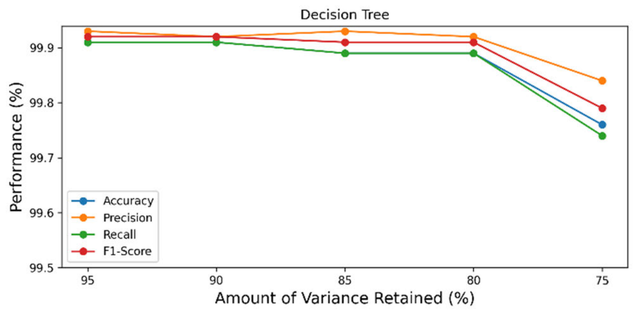

FIGURE 8. DT performance with PCA with CICIDS 2017.

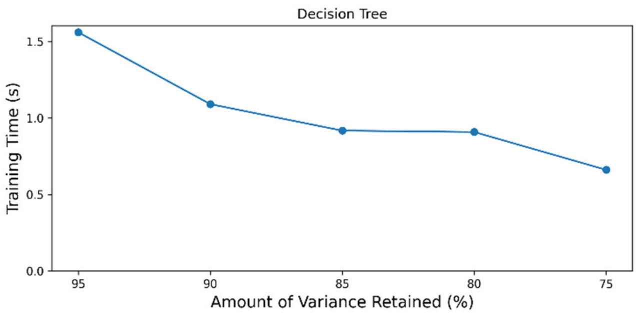

FIGURE 9. DT training time with PCA with CICIDS 2017.

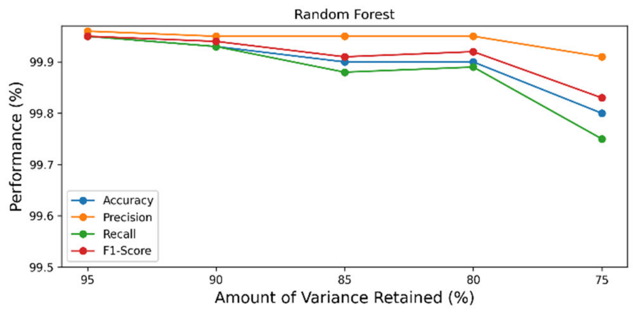

FIGURE 10. RF performance with PCA with CICIDS 2017.

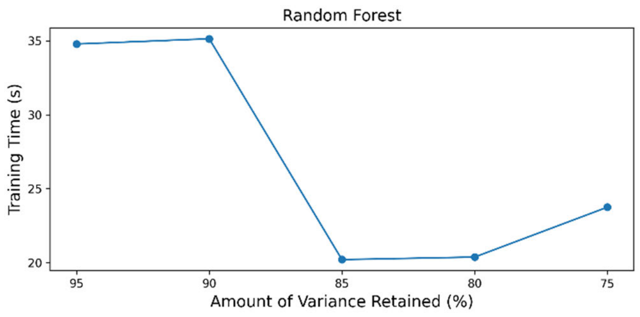

FIGURE 11. RF training time with PCA with CICIDS 2017.

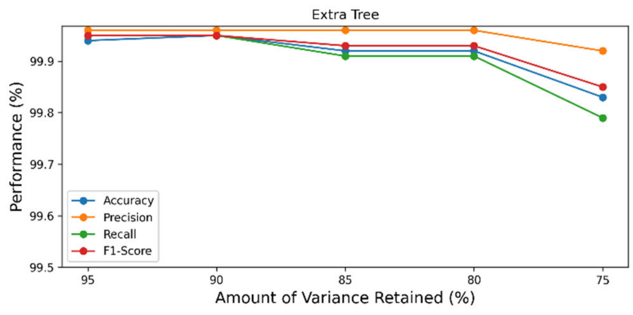

FIGURE 12. ET performance with PCA with CICIDS 2017.

variances R were 95%, 90%, 85%, 80%, and 75%, respectively. Table 4 presents the corresponding principal components, K for each R. The best results for each algorithm are highlighted in Tables 6–10. The criteria for evaluating the best results were based on two conditions: firstly, achieving high evaluation metrics and secondly, having a fast training time.

1) DT

The DT algorithm exhibited high levels of Accuracy, Recall, and F1-Score on the CICIDS 2017 dataset, with values ranging from 99.7% to 99.9%, which remained stable even when R values were modified, as shown in Figure 8. Moreover, the training time decreases as R decreases, with the fastest training time of 0.6 seconds observed at $R = 75\%$, as shown in Figure 9. We also evaluated the performance of the model on the CSE-CIC-IDS 2018 dataset. The detailed results for both datasets are presented in Table 6.

2) RF

The RF algorithm produced similarly good performance results, with metric values of approximately 99.9%, as shown in Figure 10. However, the training time of the RF algorithm does not improve. When R was 90%, the training time was the slowest, with a corresponding value of 35.1 seconds, as shown in Figure 11. Although the training time is only 20.2 seconds when R equals 85%, we have observed that the RF algorithm does not perform well in terms of Training Time when PCA is applied. Table 7 lists the specific experimental results for the RF algorithm.

3) ET

We obtained good performance results when training and testing the ET algorithm on both datasets, as shown in Figure 12. The values of these metrics exceeded 99.9%. In addition, the

TABLE 3. Features after cleaning data.

Fearure	Description	CICIDS 2017	CSE-CIC-IDS 2018
Dest Port	Destination Port	Yes	Yes
Fwd Packet Length Mean	Mean size of packet in forward direction	Yes	No
Subflow Fwd Bytes	The average number of bytes in a sub flow in the forward direction	Yes	No
Flow Duration	Duration of the flow in Microsecond	Yes	Yes
Avg Bwd Segment Size	Average size observed in the backward direction	Yes	No
Total Length of Fwd Packets	Total size of packet in forward direction	Yes	No
Active Mean	Minimum time a flow was active before becoming idle	Yes	No
Total Length of Bwd Packets	Total size of packet in backward direction	Yes	No
Fwd Packet Length Std	Standard deviation size of packet in forward direction	Yes	No
Flow IAT Std	Standard deviation time between two packets sent in the flow	Yes	Yes
Flow Packets/s	Number of flow packets per second	Yes	Yes
Flow IAT Mean	Mean time between two packets sent in the flow	Yes	Yes
Bwd Packet Length Mean	Mean size of packet in backward direction	Yes	No
Flow Bytes/s	Number of flow bytes per second	Yes	Yes
Bwd Packet Length Std	Standard deviation size of packet in backward direction	Yes	No
Flow IAT Max	Maximum time between two packets sent in the flow	Yes	Yes
Packet Length Max	Maximum length of a packet	Yes	No
Fwd IAT Total	Total time between two packets sent in the forward direction	Yes	Yes
Flow IAT Min	Minimum time between two packets sent in the flow	Yes	Yes
Fwd IAT Mean	Mean time between two packets sent in the forward direction	Yes	Yes
Avg Fwd Segment Size	Average size observed in the forward direction	Yes	No
Fwd IAT Min	Minimum time between two packets sent in the forward direction	Yes	Yes
Bwd IAT Total	Total time between two packets sent in the backward direction	Yes	Yes
Fwd IAT Std	Standard deviation time between two packets sent in the forward direction	Yes	Yes
Fwd IAT Max	Maximum time between two packets sent in the forward direction	Yes	Yes
Bwd IAT Mean	Mean time between two packets sent in the backward direction	Yes	Yes
Bwd IAT Std	Standard deviation time between two packets sent in the backward direction	Yes	Yes
Bwd IAT Min	Minimum time between two packets sent in the backward direction	Yes	Yes
Bwd IAT Max	Maximum time between two packets sent in the backward direction	Yes	Yes
Bwd Packets/s	Number of backward packets per second	Yes	Yes
Fwd Packets/s	Number of forward packets per second	Yes	Yes
Packet Length Std	Standard deviation length of a packet	Yes	No
Packet Length Mean	Mean length of a packet	Yes	No
Packet Length Variance	Variance length of a packet	Yes	No

TABLE 3. *(Continued.)* **Features after cleaning data.**

Subflow Bwd Bytes	The average number of bytes in a sub flow in the backward direction	Yes	No
Average Packet Size	Average size of packet	Yes	No
Active Std	Standard deviation time a flow was active before becoming idle	Yes	No
Active Min	Minimum time a flow was active before becoming idle	Yes	No
Active Max	Maximum time a flow was active before becoming idle	Yes	No
Idle Std	Standard deviation time a flow was idle before becoming active	Yes	No
Idle Mean	Mean time a flow was idle before becoming active	Yes	No
Idle Min	Minimum time a flow was idle before becoming active	Yes	No
Idle Max	Maximum time a flow was idle before becoming active	Yes	No

TABLE 4. **Principal components.**

R Amount of variance retained	95%	90%	85%	80%	75%
K CICIDS 2017	6	4	3	3	2
K CSE-CIC-IDS 2018	2	2	2	1	1

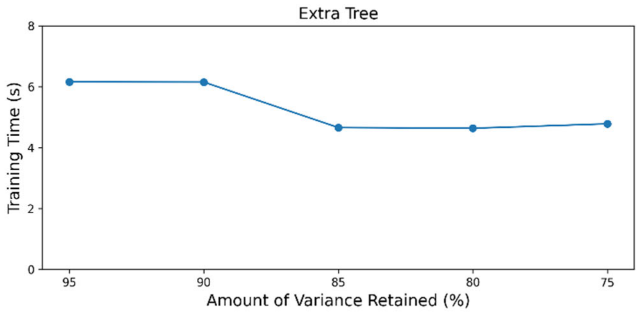

FIGURE 13. **ET training time with PCA with CICIDS 2017.**

training time improved with an increase in R. The algorithm only took 4.6 seconds to complete when R was 80% with the CICIDS 2017 dataset and 8.4 seconds when R was 85% with the CSE-CIC-IDS 2018 dataset, as shown in Figure 13. Table 8 presents the evaluation results of the proposed model using these two datasets.

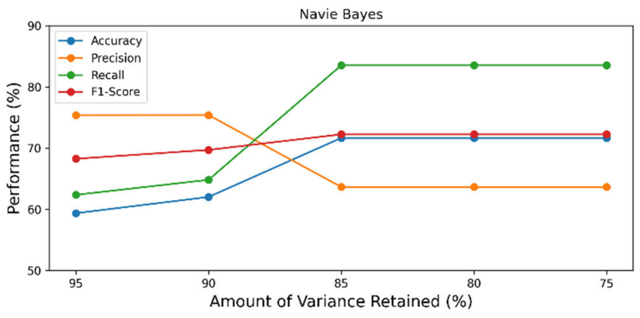

FIGURE 14. **NB performance with PCA with CICIDS 2017.**

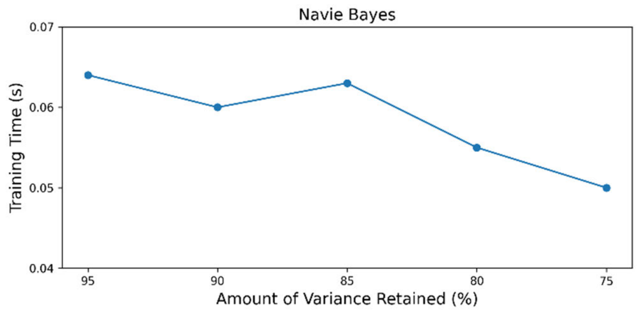

FIGURE 15. **NB training time with PCA with CICIDS 2017.**

4) NB

As mentioned in Section III, the NB algorithm typically has a fast training time, which was also true in our experiments. With the CICIDS 2017 dataset, the algorithm only took 0.05 seconds to complete the training of the model, as shown in Figure 15. However, the accuracy of the algorithm was low, with an accuracy of only 59% when R was 95%. The highest accuracy for the NB algorithm was 71% when R was 85%, as shown in Figure 14. The measurement results for the CICIDS 2017 and CSE-CIC-IDS 2018 datasets are presented in Table 9. Both datasets exhibit the best model performance when R is 75%.

TABLE 5. Detail performance without PCA on CICIDS 2017 dataset and CSE-CIC-IDS 2018 dataset.

Algorithms	CICIDS 2017 Dataset					CSE-CIC-IDS 2018 Dataset				
	Accuracy	Precision	Recall	F1-Score	Training Time (s)	Accuracy	Precision	Recall	F1-Score	Training Time (s)
DT	0.9997	0.9997	0.9997	0.9997	1.793	1	0.9999	1	1	1.913
RF	0.9997	0.9997	0.9999	0.9998	24.21	1	1	0.9999	1	45.874
ET	0.9996	0.9997	0.9997	0.9997	9.717	1	1	1	1	11.741
NB	0.8704	0.9886	0.8233	0.8984	0.23	0.9959	1	0.9885	0.9942	0.135
SVM	0.9701	0.9982	0.9525	0.9748	1,190.557	0.9998	1	0.9993	0.9997	10.091

TABLE 6. DT algorithm performance on CICIDS 2017 and CSE-CIC-IDS 2018 dataset.

Our proposed	CICIDS 2017 Dataset					CSE-CIC-IDS 2018 Dataset				
	Accuracy	Precision	Recall	F1-Score	Training Time (s)	Accuracy	Precision	Recall	F1-Score	Training Time (s)
DT_PCA_95	0.9991	0.9993	0.9991	0.9992	1.559	0.9999	0.9999	0.9999	0.9999	0.741
DT_PCA_90	0.9991	0.9992	0.9991	0.9992	1.09	0.9999	0.9999	0.9999	0.9999	0.728
DT_PCA_85	0.9989	0.9993	0.9989	0.9991	0.917	0.9999	0.9999	0.9999	0.9999	1.733
DT_PCA_80	0.9989	0.9992	0.9989	0.9991	0.908	0.9995	0.9992	0.9993	0.9993	0.653
DT_PCA_75	**0.9976**	**0.9984**	**0.9974**	**0.9979**	**0.661**	**0.9995**	**0.9992**	**0.9993**	**0.9993**	**0.406**

TABLE 7. RF algorithm performance on CICIDS 2017 dataset and CSE-CIC-IDS 2018 dataset.

Our proposed	CICIDS 2017 Dataset					CSE-CIC-IDS 2018 Dataset				
	Accuracy	Precision	Recall	F1-Score	Training Time (s)	Accuracy	Precision	Recall	F1-Score	Training Time (s)
RF_PCA_95	0.9995	0.9996	0.9995	0.9995	34.786	0.9999	0.9999	0.9999	0.9999	37.222
RF_PCA_90	0.9993	0.9995	0.9993	0.9994	35.138	**0.9999**	**0.9999**	**0.9999**	**0.9999**	**32.794**
RF_PCA_85	**0.9990**	**0.9995**	**0.9988**	**0.9991**	**20.207**	0.9999	0.9999	0.9999	0.9999	34.463
RF_PCA_80	0.9990	0.9995	0.9989	0.9992	20.377	0.9995	0.9993	0.9993	0.9993	37.488
RF_PCA_75	0.9980	0.9991	0.9975	0.9983	23.749	0.9995	0.9993	0.9993	0.9993	37.656

TABLE 8. ET algorithm performance on CICIDS 2017 dataset and CSE-CIC-IDS 2018 dataset.

Our proposed	CICIDS 2017 Dataset					CSE-CIC-IDS 2018 Dataset				
	Accuracy	Precision	Recall	F1-Score	Training Time (s)	Accuracy	Precision	Recall	F1-Score	Training Time (s)
ET_PCA_95	0.9994	0.9996	0.9995	0.9995	6.16	0.9999	0.9999	0.9999	0.9999	8.788
ET_PCA_90	0.9995	0.9996	0.9995	0.9995	6.153	0.9999	0.9999	0.9999	0.9999	8.977
ET_PCA_85	0.9992	0.9996	0.9991	0.9993	4.66	**0.9999**	**0.9999**	**0.9999**	**0.9999**	**8.458**
ET_PCA_80	**0.9992**	**0.9996**	**0.9991**	**0.9993**	**4.637**	0.9995	0.9993	0.9993	0.9993	8.937
ET_PCA_75	0.9983	0.9992	0.9979	0.9985	4.783	0.9995	0.9992	0.9993	0.9993	8.628

TABLE 9. NB algorithm performance on CICIDS 2017 dataset and CSE-CIC-IDS 2018 dataset.

Our proposed	CICIDS 2017 Dataset					CSE-CIC-IDS 2018 Dataset				
	Accuracy	Precision	Recall	F1-Score	Training Time (s)	Accuracy	Precision	Recall	F1-Score	Training Time (s)
NB_PCA_95	0.5937	0.7538	0.6237	0.6826	0.064	0.9959	1	0.9886	0.9943	0.106
NB_PCA_90	0.6202	0.7539	0.6482	0.6971	0.06	0.9959	1	0.9886	0.9943	0.099
NB_PCA_85	0.7166	0.6366	0.8352	0.7225	0.063	0.9959	1	0.9886	0.9943	0.111
NB_PCA_80	0.7166	0.6366	0.8352	0.7225	0.055	0.9959	1	0.9886	0.9943	0.089
NB_PCA_75	**0.7166**	**0.6366**	**0.8352**	**0.7225**	**0.05**	**0.9959**	**1**	**0.9886**	**0.9943**	**0.08**

5) SVM

The SVM algorithms provided Recall, Accuracy, Precision, and F1-Score with $R = 95\%$, $R = 90\%$, $R = 85\%$, and $R = 80\%$, respectively, with the same values of 96%. However, when $R = 75\%$, the performance dropped sharply to 90%, as shown in Figure 16. The training time of the SVM

TABLE 10. SVM algorithm performance on CICIDS 2017 dataset and CSE-CIC-IDS 2018 dataset.

Our proposed	CICIDS 2017 Dataset					CSE-CIC-IDS 2018 Dataset				
	Accuracy	Precision	Recall	F1-Score	Training Time (s)	Accuracy	Precision	Recall	F1-Score	Training Time (s)
SVM_PCA_95	0.9665	0.996	0.9487	0.9718	512.449	0.9996	1	0.9988	0.9994	1629.449
SVM_PCA_90	**0.9668**	**0.9961**	**0.9491**	**0.972**	**270.85**	0.9996	1	0.9988	0.9994	1596.533
SVM_PCA_85	0.9679	0.9969	0.9502	0.973	283.286	0.9996	1	0.9988	0.9994	1550.142
SVM_PCA_80	0.9679	0.9969	0.9502	0.973	370.745	0.9994	1	0.9984	0.9992	212.251
SVM_PCA_75	0.8995	0.8989	0.9256	0.912	505.548	**0.9994**	**1**	**0.9984**	**0.9992**	**211.711**

TABLE 11. Our proposed comparison with DT, RF, ET, NB, SVM algorithm on CICIDS 2017 dataset and CSE-CIC-IDS 2018 dataset.

Algorithms	CICIDS 2017 Dataset					CSE-CIC-IDS 2018 Dataset				
	Accuracy	Precision	Recall	F1-Score	Training Time	Accuracy	Precision	Recall	F1-Score	Training Time
DT	0.9997	0.9997	0.9997	0.9997	1.793	1	0.9999	1	1	1.913
DT Best Model	0.9976	0.9984	0.9974	0.9979	0.661	0.9995	0.9992	0.9993	0.9993	0.406
C of DT	- 0.21%	- 0.13%	- 0.23%	- 0.18%	+ 2.7	- 0.05%	- 0.07%	- 0.07%	- 0.07%	+ 4.7
RF	0.9997	0.9997	0.9999	0.9998	24.21	1	1	0.9999	1	45.874
RF Best Model	0.9990	0.9995	0.9988	0.9991	20.207	0.9999	0.9999	0.9999	0.9999	32.794
C of RF	- 0.07%	- 0.02%	- 0.11%	- 0.07%	+ 1.2	- 0.01%	- 0.01%	0	- 0.01%	+ 1.4
ET	0.9996	0.9997	0.9997	0.9997	9.717	1	1	1	1	11.741
ET Best Model	0.9992	0.9996	0.9991	0.9993	4.637	0.9999	0.9999	0.9999	0.9999	8.458
C of ET	- 0.04%	- 0.01%	- 0.06%	- 0.04%	+ 2.1	- 0.01%	- 0.01%	- 0.01%	- 0.01%	+ 1.4
NB	0.8704	0.9886	0.8233	0.8984	0.23	0.9959	1	0.9885	0.9942	0.135
NB Best Model	0.7166	0.6366	0.8352	0.7225	0.05	0.9959	1	0.9886	0.9943	0.08
C of NB	- 17.67%	- 35.6%	+ 1.4%	- 19.5%	+ 4.6	0	0	- 0.01%	- 0.01%	+ 1.7
SVM	0.9701	0.9982	0.9525	0.9748	1,190.557	0.9998	1	0.9993	0.9997	10.091
SVM Best Model	0.9668	0.9961	0.9491	0.972	270.85	0.9994	1	0.9984	0.9992	211.711
C of SVM	- 0.34%	- 0.21%	- 0.35%	- 0.28%	+ 4.4	- 0.04%	0	- 0.09%	- 0.05%	- 21

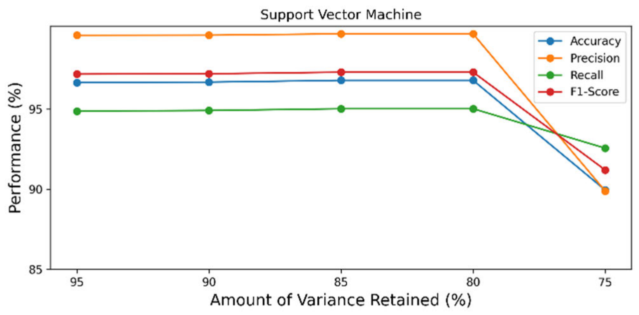

FIGURE 16. SVM performance with PCA with CICIDS 2017.

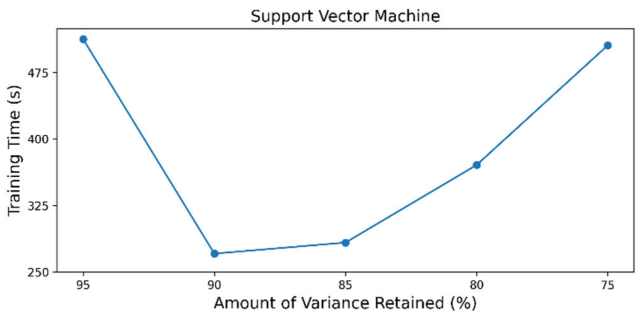

FIGURE 17. SVM training time with PCA with CICIDS 2017.

algorithm was the slowest compared to the rest of the algorithms. At $R = 90\%$, the algorithm required 270 seconds for training. However, when R was set to 95% and 75%, the training time significantly increased to 512 seconds and 505 seconds, respectively, as shown in Figure 17. The results of the SVM algorithm are presented in Table 10.

Note that in Tables 6 through 10, we named our proposed model A_PCA_R, where *A* being the name of the evaluated algorithm and *R* is the amount of variance retained after PCA. Example: SVM_PCA_95 indicates that our proposed model uses the SVM algorithm with a retained variance, *R* of 95% after applying PCA.

E. COMPARISON RESULTS

To aid in the evaluation of our proposed model, we compared evaluation metrics such as Accuracy, Precision, Recall, F1-Score, and Training Time between the best-performing proposed model and the corresponding ML algorithms without applying PCA. We used a quantity called the change magnitude C to demonstrate comprehensive improvement in our approach, thereby helping us achieve the purpose of our study. We conducted a thorough analysis of the results, and the following details provided more specific information.

1) ACCURACY, PRECISION, RECALL, F1-SCORE

The proposed models achieved an accuracy of approximately 99.9%. Most of them showed a small reduction rate, ranging from 0.01% to 0.34%, in evaluation metrics such as Accuracy, Precision, Recall, and F1-Score for both datasets. This reduction rate indicates that the accuracy of our proposed model is comparable to that of ML algorithms when PCA is not applied. The $C_{Accuracy}$ for the DT, RF, ET, NB, and SVM algorithms on the CSE-CIC-IDS 2018 dataset were -0.05%, -0.01%, -0.01%, 0%, and -0.04%, respectively. One notable case is the NB algorithm, which showed a significant decrease of 17% in accuracy when evaluated using the CICIDS 2017 dataset. We conducted a detailed comparison of our proposed method, as shown in Table 11.

2) TRAINING TIME

The training time of the proposed model was significantly improved. Specifically, the DT, RF, ET, NB, and SVM algorithms were improved by factors of 2.7, 1.2, 2.1, 4.6, and 4.4, respectively, when evaluated using the CICIDS 2017 dataset. For the CSE-CIC-IDS 2018 dataset, improvements were made in the DT, RF, ET, and NB algorithms. However, the training time of the SVM algorithm was increased by a factor of 21. The evaluation metrics are listed in Table 11. Note that the symbol (-) indicates a decrease and the symbol (+) indicates an improvement in the C evaluation metrics. DT_PCA_75, RF_PCA_85, ET_PCA_80, NB_PCA_75, and SVM_PCA_90 were the best models for the CICIDS 2017 dataset, whereas DT_PCA_75, RF_PCA_90, ET_PCA_85, NB_PCA_75, and SVM_PCA_75 were the best for the CSE-CIC-IDS 2018 dataset.

VI. CONCLUSION

The IoT is currently and will continue to be one of the leading fields facilitating social development. Ensuring the stability and security of IoT systems is the top priority. Previous studies have mostly focused on powerful servers in traditional computer networks or overlooked the importance of the training time for the model, only emphasizing its precision. Consequently, these studies are not suitable for IoT systems because of the limited resources available on the edge devices. In this study, we develop a model that can be deployed on edge IoT devices to detect DDoS attacks. We combined the ML algorithms DT, RF, ET, NB, and SVM with PCA. We then evaluated our proposed model on two popular network-based datasets, CICIDS 2017 and CSE-CIC-IDS 2018. The results showed that our proposed model performed well in predicting DDoS attacks and improved model training time. Specifically, we found that the proposed models based on DT and ET algorithms achieved a prediction accuracy threshold of 99.99% and training times three times and two times faster, respectively, when compared to the algorithms without PCA.

In the future, we plan to evaluate our proposed model on real-world datasets and to develop it for practical IoT applications. We will continue to research an approach to improve the multiclass classification capabilities of our model.

REFERENCES

[1] D. Velasquez, E. Perez, X. Oregui, A. Artetxe, J. Manteca, J. E. Mansilla, M. Toro, M. Maiza, and B. Sierra, "A hybrid machine-learning ensemble for anomaly detection in real-time industry 4.0 systems," *IEEE Access*, vol. 10, pp. 72024–72036, 2022.

[2] S. U. Rehman and V. Gruhn, "An approach to secure smart homes in cyber-physical systems/Internet-of-Things," in *Proc. 5th Int. Conf. Softw. Defined Syst. (SDS)*, Barcelona, Spain, Apr. 2018, pp. 126–129.

[3] S. K. Vishwakarma, P. Upadhyaya, B. Kumari, and A. K. Mishra, "Smart energy efficient home automation system using IoT," in *Proc. 4th Int. Conf. Internet Things, Smart Innov. Usages (IoT-SIU)*, Ghaziabad, India, Apr. 2019, pp. 417–420.

[4] S. Chaudhary, R. Johari, R. Bhatia, K. Gupta, and A. Bhatnagar, "CRAIoT: Concept, review and application(s) of IoT," in *Proc. 4th Int. Conf. Internet Things, Smart Innov. Usages (IoT-SIU)*, Ghaziabad, India, Apr. 2019, pp. 402–405.

[5] (2022). *Lionel Sujay Vailshery*. [Online]. Available: https://www.statista.com

[6] N. Mishra and S. Pandya, "Internet of Things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review," *IEEE Access*, vol. 9, pp. 59353–59377, 2021.

[7] *X-Force Threat Intelligence Index 2022*, IBM Security, Atlanta, GA, USA, 2022.

[8] D. Patel, "A study on DDOS attacks, danger and its prevention," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 10, no. 12, pp. 1962–1967, Dec. 2022.

[9] N. Vlajic and D. Zhou, "IoT as a land of opportunity for DDoS hackers," *Computer*, vol. 51, no. 7, pp. 26–34, Jul. 2018.

[10] T. U. Sheikh, H. Rahman, H. S. Al-Qahtani, T. K. Hazra, and N. U. Sheikh, "Countermeasure of attack vectors using signature-based IDS in IoT environments," in *Proc. IEEE 10th Annu. Inf. Technol., Electron. Mobile Commun. Conf. (IEMCON)*, Vancouver, BC, Canada, Oct. 2019, pp. 1130–1136.

[11] R. Zhang, J.-P. Condomines, N. Larrieu, and R. Chemali, "Design of a novel network intrusion detection system for drone communications," in *Proc. IEEE/AIAA 37th Digit. Avionics Syst. Conf. (DASC)*, London, U.K., Sep. 2018, pp. 241–250.

[12] F. Suthar, N. Patel, and S. V. O. Khanna, "A signature-based botnet (Emotet) detection mechanism," *Int. J. Eng. Trends Technol.*, vol. 70, no. 5, pp. 185–193, May 2022.

[13] A. M. da Silva Cardoso, R. F. Lopes, A. S. Teles, and F. B. V. Magalhaes, "Poster abstract: Real-time DDoS detection based on complex event processing for IoT," in *Proc. IEEE/ACM 3rd Int. Conf. Internet-Things Design Implement. (IoTDI)*, Orlando, FL, USA, Apr. 2018, pp. 273–274.

[14] M. Dimolianis, A. Pavlidis, and V. Maglaris, "Signature-based traffic classification and mitigation for DDoS attacks using programmable network data planes," *IEEE Access*, vol. 9, pp. 113061–113076, 2021.

[15] A. Praseed and P. S. Thilagam, "HTTP request pattern based signatures for early application layer DDoS detection: A firewall agnostic approach," *J. Inf. Secur. Appl.*, vol. 65, Mar. 2022, Art. no. 103090.

[16] X. You, Y. Feng, and K. Sakurai, "Packet in message based DDoS attack detection in SDN network using OpenFlow," in *Proc. 5th Int. Symp. Comput. Netw. (CANDAR)*, Nov. 2017, pp. 522–528.

[17] K. Wehbi, L. Hong, T. Al-salah, and A. A. Bhutta, "A survey on machine learning based detection on DDoS attacks for IoT systems," in *Proc. SoutheastCon*, Huntsville, AL, USA, Apr. 2019, pp. 1–6.

[18] Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa, and C. F. M. Foozy, "Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset," *IEEE Access*, vol. 9, pp. 22351–22370, 2021.

[19] M. Najafimehr, S. Zarifzadeh, and S. Mostafavi, "A hybrid machine learning approach for detecting unprecedented DDoS attacks," *J. Supercomput.*, vol. 78, no. 6, pp. 8106–8136, Apr. 2022.
[20] D. Erhan and E. Anarim, "Hybrid DDoS detection framework using matching pursuit algorithm," *IEEE Access*, vol. 8, pp. 118912–118923, 2020.
[21] A. Praseed and P. S. Thilagam, "Modelling behavioural dynamics for asymmetric application layer DDoS detection," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 617–626, 2021.
[22] J. Li, Z. Xue, C. Li, and M. Liu, "RTED-SD: A real-time edge detection scheme for Sybil DDoS in the Internet of Vehicles," *IEEE Access*, vol. 9, pp. 11296–11305, 2021.
[23] Z. Liu, X. Yin, and Y. Hu, "CPSS LR-DDoS detection and defense in edge computing utilizing DCNN Q-learning," *IEEE Access*, vol. 8, pp. 42120–42130, 2020.
[24] A. Alsirhani, S. Sampalli, and P. Bodorik, "DDoS detection system: Using a set of classification algorithms controlled by fuzzy logic system in Apache spark," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 3, pp. 936–949, Sep. 2019.
[25] F. Alasmary, S. Alraddadi, S. Al-Ahmadi, and J. Al-Muhtadi, "Shield-RNN: A distributed flow-based DDoS detection solution for IoT using sequence majority voting," *IEEE Access*, vol. 10, pp. 88263–88275, 2022.
[26] M. S. E. Sayed, N.-A. Le-Khac, M. A. Azer, and A. D. Jurcut, "A flow-based anomaly detection approach with feature selection method against DDoS attacks in SDNs," *IEEE Trans. Cognit. Commun. Netw.*, vol. 8, no. 4, pp. 1862–1880, Dec. 2022.
[27] H. Beitollahi, D. M. Sharif, and M. Fazeli, "Application layer DDoS attack detection using cuckoo search algorithm-trained radial basis function," *IEEE Access*, vol. 10, pp. 63844–63854, 2022.
[28] V. Deepa, K. M. Sudar, and P. Deepalakshmi, "Design of ensemble learning methods for DDoS detection in SDN environment," in *Proc. Int. Conf. Vis. Towards Emerg. Trends Commun. Netw. (ViTECoN)*, Vellore, India, Mar. 2019, pp. 1–6.
[29] Y. N. Soe, P. I. Santosa, and R. Hartanto, "DDoS attack detection based on simple ANN with SMOTE for IoT environment," in *Proc. 4th Int. Conf. Informat. Comput. (ICIC)*, Semarang, Indonesia, Oct. 2019, pp. 1–5.
[30] Y. Jia, F. Zhong, A. Alrawais, B. Gong, and X. Cheng, "FlowGuard: An intelligent edge defense mechanism against IoT DDoS attacks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9552–9562, Oct. 2020.
[31] S. P. Khedkar and R. AroulCanessane, "Machine learning model for classification of IoT network traffic," in *Proc. 4th Int. Conf. I-SMAC*, Palladam, India, Oct. 2020, pp. 166–170.
[32] M. Zeeshan, Q. Riaz, M. A. Bilal, M. K. Shahzad, H. Jabeen, S. A. Haider, and A. Rahim, "Protocol-based deep intrusion detection for DoS and DDoS attacks using UNSW-NB15 and bot-IoT data-sets," *IEEE Access*, vol. 10, pp. 2269–2283, 2022.
[33] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116.
[34] B. Tushir, Y. Dalal, B. Dezfouli, and Y. Liu, "A quantitative study of DDoS and E-DDoS attacks on WiFi smart home devices," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6282–6292, Apr. 2021.
[35] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.
[36] V. Hnamte and J. Hussain, "DCNNBiLSTM: An efficient hybrid deep learning-based intrusion detection system," *Telematics Informat. Rep.*, vol. 10, Jun. 2023, Art. no. 100053.
[37] O. D. Okey, D. C. Melgarejo, M. Saadi, R. L. Rosa, J. H. Kleinschmidt, and D. Z. Rodriguez, "Transfer learning approach to IDS on cloud IoT devices using optimized CNN," *IEEE Access*, vol. 11, pp. 1023–1038, 2023.
[38] Y. Wei, J. Jang-Jaccard, F. Sabrina, A. Singh, W. Xu, and S. Camtepe, "AE-MLP: A hybrid deep learning approach for DDoS detection and classification," *IEEE Access*, vol. 9, pp. 146810–146821, 2021.
[39] S. Haider, A. Akhunzada, I. Mustafa, T. B. Patel, A. Fernandez, K.-K.-R. Choo, and J. Iqbal, "A deep CNN ensemble framework for efficient DDoS attack detection in software defined networks," *IEEE Access*, vol. 8, pp. 53972–53983, 2020.
[40] Z. Liu, C. Guo, D. Liu, and X. Yin, "An asynchronous federated learning arbitration model for low-rate DDoS attack detection," *IEEE Access*, vol. 11, pp. 18448–18460, 2023.
[41] M. Roopak, G. Y. Tian, and J. Chambers, "Deep learning models for cyber security in IoT networks," in *Proc. IEEE 9th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Las Vegas, NV, USA, Jan. 2019.
[42] M. Ghurab, G. Gaphari, F. Alshami, R. Alshamy, and S. Othman, "A detailed analysis of benchmark datasets for network intrusion detection system," *Asian J. Res. Comput. Sci.*, vol. 2021, pp. 14–33, Apr. 2021.
[43] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "An evaluation framework for intrusion detection dataset," in *Proc. Int. Conf. Inf. Sci. Secur. (ICISS)*, Pattaya, Thailand, Dec. 2016, pp. 41–45.

NGUYEN TAN CAM received the bachelor's degree in information technology and the master's degree in information systems from the University of Natural Sciences, Vietnam National University Ho Chi Minh City, in 2006 and 2010, respectively, and the Ph.D. degree in information technology from the University of Information Technology, Vietnam National University Ho Chi Minh City (UIT VNU-HCM), under the supervision of Dr. Tuan Nguyen and Dr. Van-Hau Pham. He is currently a Lecturer with UIT VNU-HCM. His main research interests include mobile devices, networks, and the IoT security.

NGUYEN GIA TRUNG received the bachelor's degree in information technology from Nong Lam University, in 2013. He is currently a Researcher with UIT VNU-HCM. His main research interests include intrusion detection, network security, machine learning, and the Internet of Things security.