



A hybrid machine learning approach for detecting unprecedented DDoS attacks

Mohammad Najafimehr¹ · Sajjad Zarifzadeh¹ · Seyedakbar Mostafavi¹

Accepted: 16 December 2021 / Published online: 7 January 2022

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract

Service availability plays a vital role on computer networks, against which Distributed Denial of Service (DDoS) attacks are an increasingly growing threat each year. Machine learning (ML) is a promising approach widely used for DDoS detection, which obtains satisfactory results for pre-known attacks. However, they are almost incapable of detecting unknown malicious traffic. This paper proposes a novel method combining both supervised and unsupervised algorithms. First, a clustering algorithm separates the anomalous traffic from the normal data using several flow-based features. Then, using certain statistical measures, a classification algorithm is used to label the clusters. Employing a big data processing framework, we evaluate the proposed method by training on the CICIDS2017 dataset and testing on a different set of attacks provided in the more up-to-date CICDDoS2019. The results demonstrate that the Positive Likelihood Ratio (LR+) of our method is approximately 198% higher than the ML classification algorithms.

Keywords DDoS detection · Machine learning · DBSCAN · Network security · Big data · Unprecedented attacks

1 Introduction

Distributed Denial of Service (DDoS) attack involves malicious techniques to diminish the availability of services in computer networks, for which the most prevalent way is sending massive traffic toward the target to exhaust either the bandwidth or

✉ Sajjad Zarifzadeh
sjarifzadeh@yazd.ac.ir

Mohammad Najafimehr
mohammad.najafimehr@stu.yazd.ac.ir

Seyedakbar Mostafavi
a.mostafavi@yazd.ac.ir

¹ Department of Computer Engineering, Yazd University, Yazd, Iran

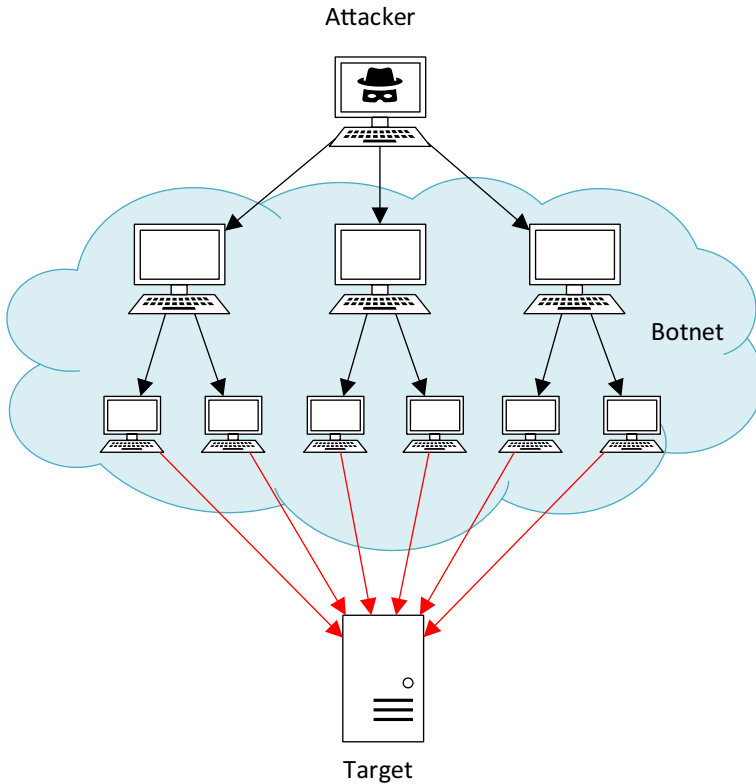


Fig. 1 A diagram of a DDoS attack performed with a botnet

the target's resources. Attackers usually use many computers, which are called bots or zombies, to transfer malicious traffic. The bots are often the computers compromised by attackers, while their legitimate controllers are unaware of that [5]. Figure 1 indicates a basic diagram of how a DDoS attack occurs, in which the attacker controls a botnet to employ the attack on the target. In the contemporary techniques for the attack, the attacker forges the target's IP address and sends requests to servers across the Internet; consequently, they respond to the target with a large amount of traffic. This type of attack is called a reflection attack, and the servers are called reflectors. Reflection attacks are usually amplified, called amplification attacks, in which the size of the response traffic is much larger than the requests' size sent by the attacker. Attackers may launch this type of attack by sending a tiny request, querying a list of information. For instance, NetBIOS DDoS is an amplified attack with an amplification factor of 3.8 [47], which means that the response size is 3.8 times as much as the request size. It is crucial to defend against DDoS attacks, owing to the depletion of services' availability that not only troubles legitimate users but also may threaten human lives concerning medical devices connected to the network in an Internet of Things (IoT) environment [54]. Additionally, this type of attack is

increasing year by year. According to the Kaspersky reports [28, 29], the number of DDoS attacks in 2020 compared to 2019 and 2018 has grown by approximately 88% and 121%, respectively. Furthermore, as reported by Imperva [26], due to the COVID-19 outbreak, lockdowns, and consequently changing people and businesses to online activities, DDoS attacks has been about 4.3 times¹ as much magnitude as before the pandemic.

Among many defense mechanisms proposed for detecting DDoS attacks, Machine learning (ML) methods, which have been demonstrated to be helpful in cybersecurity [55], are also used to address this challenge. ML-based DDoS detection methods may be categorized into three categories: supervised, unsupervised, and hybrid methods.

The supervised methods use a labeled dataset, where the records' labels are specified as a column called "class label." The datasets are mostly flow-based, meaning that every record indicates a network flow, i.e., the packets that share the same source IP, source port, destination IP, destination port, and timestamp. However, some datasets may be packet-based, i.e., each record of them indicates a network packet. The unsupervised methods use the records without any labels; they are usually utilized for clustering the traffic to separate the anomalous traffic from the normal traffic. The hybrid methods often attempt to combine supervised and unsupervised methods to overcome their shortcomings and also provide their benefits.

Many ML-based approaches are proposed to detect DDoS attacks, and it has been reported that they are satisfactorily accurate as it is discussed in Sects. 2 and 5.4. On the other hand, not only are the DDoS attacks increasing in frequency, but they are also going to be more complicated, sophisticated, and difficult to detect [21, 27, 51, 58], while new types are emerging [11, 42, 43].

We evaluated multiple ML conventional classification algorithms by training on a dataset but testing on another more-up-to-date dataset containing more novel and different types of attack. The ML algorithms we have used are Decision Tree (DT), Random Forest (RF), Naïve Bayes (NB), and Support Vector Machine (SVM). The results show that the models are utterly incapable of detecting the novel attacks provided in the test dataset. To address this issue, we propose a hybrid ML-based approach in which a dataset of network flows is clustered using the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm [17]. Afterward, multiple statistical measures are calculated and selected as the features for the classification of clusters. Finally, each cluster is labeled whether "DDoS" or "Benign" using a classification algorithm. In the end, the predicted label of each cluster is assigned to all the flows inside. We evaluate the proposed method by training it on the CICIDS2017 dataset and testing it on the newer CICDDoS2019 dataset. We utilize multiple classification algorithms as well as some other parameters to implement our method and compare it with the conventional classification algorithms used in many other published papers. Furthermore, we evaluate the classification algorithms using a big data framework (i.e., Apache Spark [59]) on a cluster of two machines, consisting of 8 cores in total (4 cores in each machine). The evaluation

¹ From 217 Gbps in Sep. 2019 to 937 Gbps in Sep. 2020.

results showed that the accuracy, recall, precision, false positive rate (FPR), and positive likelihood ratio (LR+) of the tuned proposed method are, respectively, 14.65, 20.47, 1.35, 0.98, and 7.27 times as much as the average of the conventional classification algorithms.

The contributions of this paper are (1) discovering a pattern to distinguish between the DDoS and normal data, (2) proposing a method combined of supervised and unsupervised approaches to detect this pattern, and (3) assessing the effectiveness of the method against the data of unprecedented attack types, from a different network environment, and with the dissimilar distribution. In terms of LR+, on average, our method detects unprecedented DDoS attacks 5.89 times as effective as the conventional classification algorithms. Our observations imply that neither supervised nor unsupervised methods can provide sufficient efficacy when they are tested on a different dataset than they are trained on.

The rest of the paper is organized as follows. Section 2 discusses a summary of some related approaches proposed in recent years, which we have used in this work as the background knowledge. Section 3 discusses the observations in the DDoS data which are the idea behind our proposed method. Section 4 explains the method that we propose for the effective detection of novel and unprecedented DDoS attacks. Section 5 discusses the evaluation results. Finally, Sect. 6 concludes the paper.

2 Related work

There are many ML-based approaches proposed for DDoS detection, including supervised, unsupervised, and hybrid methods, which combines both former approaches. This section provides a summary of some recently proposed methods in this area.

2.1 Supervised and unsupervised methods

Supervised methods involve using classification algorithms to detect the attack. That is done by training a model with the aim of labeled data of many samples. This data may be network flows or packets that have already been identified as malicious or benign. The great advantage of this approach is to accurately detect the known attacks, i.e., the attacks that are used to train the model. However, they may usually not be able to detect unknown attacks.

Doshi et al. [14] implemented multiple supervised learning algorithms to detect DDoS attacks in an IoT network. They evaluated their implementations on a physical testbed by launching the HTTP Flood, SYN Flood, and UDP Flood attacks. They used some features such as *Packet size*, *Protocol*, *Bandwidth*, and the *number of distinct destination IP addresses*. They implemented DT, RF, SVM, k-Nearest Neighbors (kNN), and Artificial Neural Network (ANN) classification algorithms, and their reported result shows that the recall, precision, and accuracy of all the algorithms are approximately 0.99.

Balkanli et al. [8] implemented a couple of classification algorithms and compared their result with two IDSs using “CAIDA” dataset [1] collected from backscatter traffic. Their classification algorithms are DT and NB models constructed using some features such as *Protocol*, *SYN Flag*, and *Packet Length*. The evaluation result shows that the accuracy, precision, and recall of the DT algorithm is more appropriate than the NB; for instance, the maximum recall of the DT and NB has been reported 1 and 0.82, respectively.

Roempluk and Surinta [44] evaluated multiple supervised ML algorithms on two prevalent datasets; “KDDCup99” [10] and “NSL-KDD” [53]. The used algorithms are kNN, SVM with linear kernel, and also nonlinear and multi-layer perceptron (MLP): a type of ANN. Overall, based on accuracy, the only reported evaluation metric, the kNN with 99.99%, is the most optimal compared to the MLP and the SVM.

Rahman et al. [41] used an SDN environment simulated by Mininet [2] to evaluate DT, RF, SVM, and kNN algorithms. They used the *hping3* tool to produce the attack traffic (ICMP Flood and TCP Flood attacks) as well as the normal traffic toward the targets. Afterward, the training of the classification algorithms is performed inside the targets’ machines with the aim of the WEKA tool. As the reported results show, the F1-score, recall, accuracy, and precision of all the mentioned classifiers are equal to 1, and the DT has the minimum average of training and testing times while the kNN has the maximum.

Wani et al. [56] utilized some ML-based DDoS detection algorithms in a cloud environment. They used a computer with Kali Linux as the attacker machine and multiple zombies to launch an attack on the ownCloud platform. Extracting the dataset from the traffic and detecting the attack on the cloud are done using SNORT. The considered class label in the dataset includes two distinct values: normal, suspicious. The used classification algorithms are SVM, RF, and NB with 99.8%, 99.3%, and 86.0% of recall, respectively; hence, it is inferred that SVM has the most appropriate detection capability among others.

Morfino and Rampone [37] used multiple supervised learning algorithms, and proposed a near-real-time intrusion detection system for IoT environment, and evaluated it using a public dataset [36]. The algorithms used are DT, RF, SVM, Logistic Regression (LR), and Gradient Boosting Tree (GBT). They performed the SYN flood attack and implemented their proposed method using the Apache Spark framework. As the results show, the RF has the maximum accuracy for detecting SYN flood attacks among other algorithms.

Bakker et al. [7] first used the RF, kNN, and SVM algorithms on the publicly available ISCX dataset [52] and then evaluated them on a physical Software Defined Network (SDN) testbed. The reported F1-score of RF, kNN, and SVM are approximately 95%, 94%, and 93%, respectively. Although all the algorithms obtained satisfactory results from testing on the datasets, they yielded poor results on the physical SDN. By changing the evaluation environment from the datasets to the physical testbed, the accuracy and FPR have remained acceptable, but the recall has decreased significantly. The recall of the SVM, kNN, and RF are reported 14%, 0.02%, and 0.005%, respectively, for the SDN testbed. According to the results, true negatives are still high, but true positives have been reduced. We may infer that on the

physical network, the classification models have recognized appropriately benign traffic, even though they have falsely detected most of the attack traffic as benign. As they reported, the reduction in recall of the algorithms on the physical testbed is due to packet loss; the main reason for the observed degradation of the algorithms' recall in the physical environment.

Liang and Znati [32] proposed a deep-learning-based method using long short-term memory (LSTM), which is a recurrent ANN, and evaluated it by CICIDS2017. They trained the model on a subset of the dataset and tested on another part, which contains different types of attack, interchangeably. Thus, the network environments from which the test and train data are collected are the same. In the first experiment, the recall of their proposed method is about 300% as much as the average of the DT, SVM, and ANN, but the precision is reduced by roughly 6%. In the second experiment, where the train and test datasets are exchanged, the recall increased by approximately 59% and the precision decreased about 25%.

Yang et al. [57] proposed a method using the autoencoder, a type of unsupervised ANN typically used for dimensionality reduction. In an experiment, they trained the method using a dataset, a different subset of the CICIDS2017 than what we use, then tested it on a synthetic dataset produced by network simulation. This experiment shows a recall and FPR of 100%, i.e., the model recognizes any flows as DDoS. However, the average recall of several conventional learning algorithms is 99.66%, which is 0.34% lower, and FPR is 84.94%, which has been increased by 17.7% in their proposed method. In another experiment, the training is based on the synthetic dataset, while the test dataset comprises the same normal samples as the ones used for training, but all the attack samples in the CICIDS2017 are used as the DDoS samples. These attack types are all HTTP-based denial of service. In this case, the result is promising, 99.99% for the recall and 0.49% for the FPR, while these are, respectively, 1.60% and 0% for the DT classifier. Hence, their proposed method has a recall of about 61 times higher than the DT classifier.

2.2 Hybrid methods

There are some methods to detect the attack by benefiting from different approaches. For example, some methods use entropy analysis accompanied by ML techniques since, during a DDoS attack, the entropy of the incoming traffic usually changes and goes out of the normal range. Moreover, combining supervised and unsupervised ML approaches is also another way to precisely detect the attack. For instance, a clustering algorithm is applied, and then the classification method can detect the attack using the clustered data.

Idhammad et al. [25] proposed a hybrid learning approach to detect DDoS attacks. Their method comprises an entropy calculating step, a co-clustering step, and a classification algorithm at the end of their procedure. First, using a time window, the average entropy of four features for each record of the dataset is calculated. The features they used are *Source packet count*, *Destination packet count*, *Source byte count*, and *Destination byte count*. If the entropy values are out of a specified range, the traffic is considered suspicious. The next step is co-clustering

the traffic into three clusters and computing the information gain for each, and then the cluster having the minimum gain is regarded as normal, while the others are suspicious. Finally, they have used the Extra-Trees ensemble classifier to detect the DDoS traffic. They evaluated their approach using “NSL-KDD,” “ISCXIDS2012,” and “UNSW-NB15” [38] datasets. According to the reported results, the average accuracy and FPR of the proposed method from the three mentioned datasets are approximately 0.77 and 0.40, respectively.

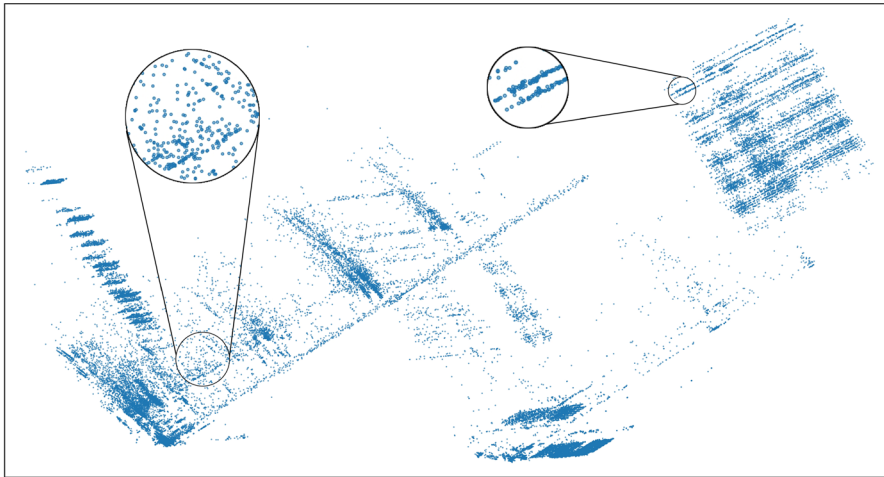
Deepa et al. [13] combined SVM, which is a supervised method, and self-organized map (SOM) as an unsupervised method. The SOM is an ANN that is used to reduce the dimension of data. According to this method, if the SVM model recognizes that connection is malicious, it will be blocked; otherwise, it is forwarded to the SOM to decide. The evaluation is conducted in a simulated SDN testbed using Scapy for producing the attack traffic. As the results show, utilizing SVM with SOM provides roughly 5% and 50% of more recall and FPR, respectively, compared to using only one of them.

Li et al. [31] proposed a real-time method that uses entropy analysis and ANN to detect high-rate DDoS attacks. It is expected that the entropy of source IP increases during a DDoS attack because of the large botnet. However, that may not be observed in practice since certain factors such as the number of targets or policies may intervene. To reduce these unwanted effects, they proposed a joint entropy by combining the entropy of source and destination IP addresses. The joint entropy of the network flows is computed in real-time through a sliding window. A predicted value of entropy is calculated by LSTM and then is deducted from the true measured value to lessen the effect of noise and jitter. By monitoring entropy changes through the sliding window, the method is able to detect the attack. The evaluation is done on “1999 DARPA,” “2009 DARPA,” and “CICDDoS2019” datasets in addition to a synthetic from a simulating an SDN testbed.

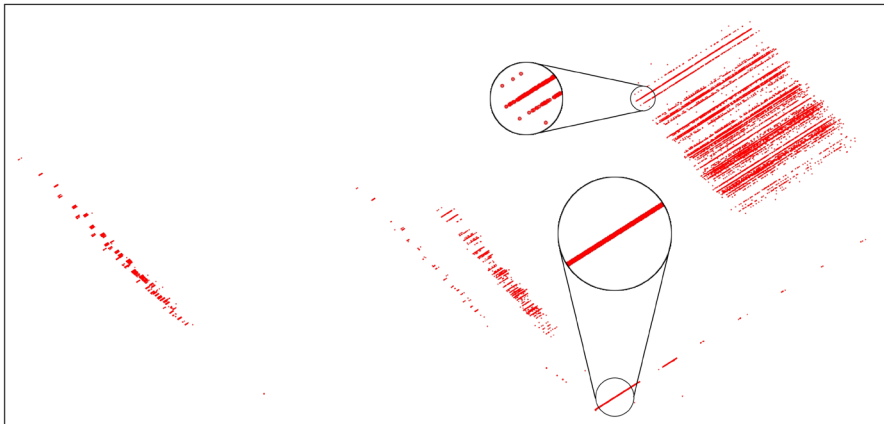
3 Observations

To get a better insight about the data, we reduced the dimensions of the CICIDS2017 dataset using the principal component analysis (PCA) algorithm into two dimensions; hence, it can be visualized in a scatter plot as shown in Fig. 2. Figure 2a shows all the Benign points, which their number is 97,718. On the other hand, all the DDoS points are shown in Fig. 2b, which are 128,027. The scatter plot can help us realize the distribution of the data in two DDoS and Benign classes. From the plot, we can infer two following observations:

1. The benign points have a more tendency to be scattered, while the DDoS ones are mostly compact and form dense areas.
2. In addition to the density of the DDoS points, it appears that they are relatively lined up.



(a) Benign

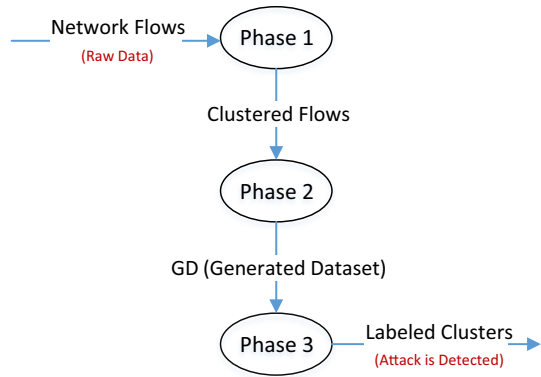


(b) DDoS

Fig. 2 The scatter plot of the data points in CICIDS2017

More to the point, while the DDoS points are about 30% higher than the Benign ones in frequency, it appears in the plot that they are fewer. The reason behind this issue is that a vast majority of the DDoS points are located on top of each other, demonstrating the high density of them. From the above observations, we can conclude that there is a special pattern in the data, which can assist with distinguishing between benign and attack network flows. In the next section, we will introduce our proposed method to find this pattern to detect the DDoS attack.

Fig. 3 Input and output of each phase of the proposed method



4 The proposed method

This section describes the proposed method for DDoS detection, which comprises three phases. Phase 1 (unsupervised phase) involves an unsupervised learning method (clustering) to separate the DDoS traffic from the benign traffic; hence, as Fig. 3 shows, it takes the network flows as the raw data and then outputs the clustered flows to phase 2. However, it cannot determine that a cluster contains [mostly] DDoS traffic or benign traffic. Therefore, a classifier is needed to label the clusters leading to the use of phases 2 and 3. As shown in the figure, the clustered flows are given as input in phase 2 (cluster analyzing phase), and subsequently, several statistical measures are calculated for each cluster which are then regarded as the features for the classification algorithm in phase 3 (supervised phase). Hence, the middle phase results in the production of a novel dataset called Generated Dataset (GD), in which every record indicates a cluster, i.e., each record in GD indicates the mentioned features computed for each cluster. Finally, phase 3 takes the GD (clusters' information), and then the classification algorithm is used to label the clusters, i.e., to determine whether a cluster includes [mostly] the benign traffic or the attack traffic. Algorithms 1 and 2 show the pseudocode for the training and testing steps of the proposed method, respectively, where lines 1-3 stand for phase 1, lines 4-9 in Algorithm 1 and 4-8 in Algorithm 2 indicate phase 2, and lines 10-11 in Algorithm 1 and 9-11 in Algorithm 2 show phase 3. Moreover, Fig. 4 shows the architecture and procedure of the proposed method as well as an example, which we discuss more below.

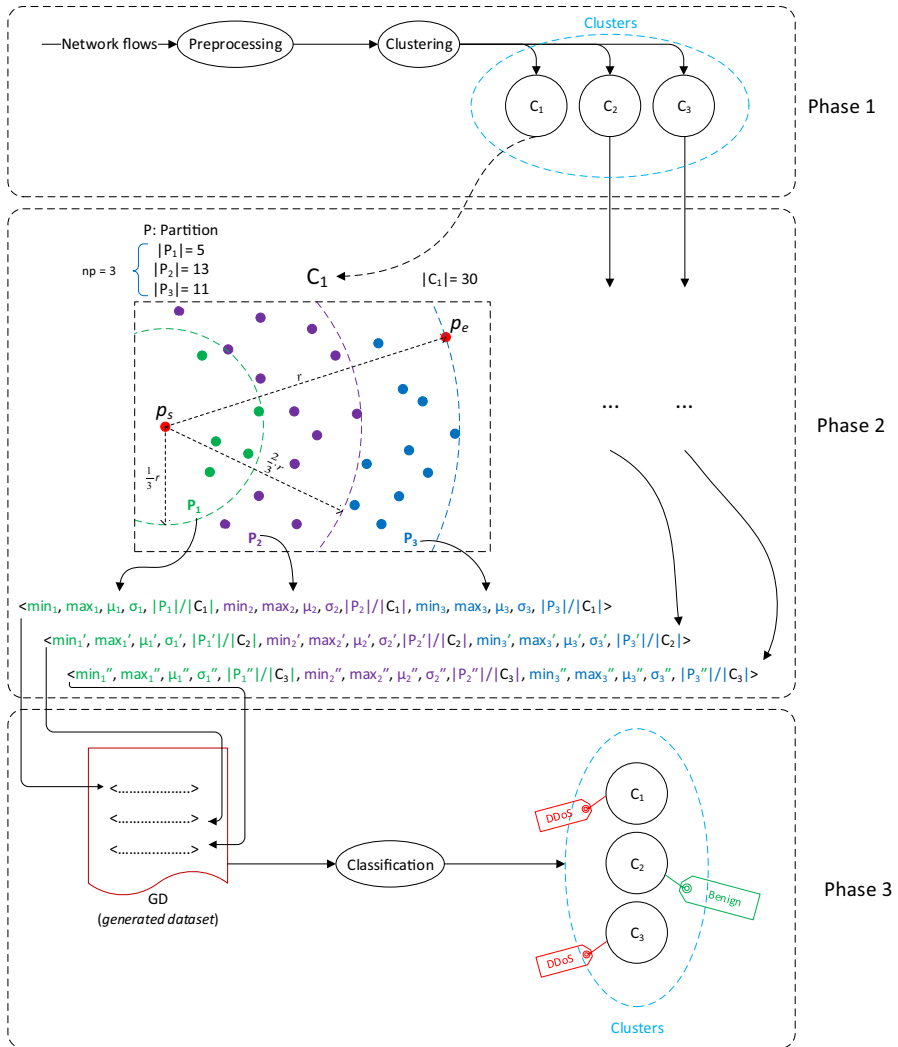


Fig. 4 Architecture of the proposed method illustrated in an example where 3 clusters are obtained

Algorithm 1 Pseudocode for the training step of the proposed method**Input:** A dataset D of network flows**Output:** A trained ML model

```

1: preprocess( $D$ )
2:  $\varepsilon \leftarrow \text{findOptimalEpsilon}()$ 
3:  $\text{clusters} \leftarrow \text{DBSCAN}(\text{Dataset}, \varepsilon, \text{minPts})$ 
4:  $GD$ , global variable ▷ generated dataset
5: for each cluster  $C$  in  $\text{clusters}$  do
6:    $\text{row} = \text{analyzeCluster}(C)$ 
7:   append  $|\text{DDoS}|/|C|$  to  $\text{row}$ 
8:   emit  $\text{row}$  into  $GD$  and reset  $\text{row}$  value
9: end for
10:  $\text{model} = \text{trainClassifier}(GD)$ 
11: save( $\text{model}$ )

```

Algorithm 2 Pseudocode for the testing step of the proposed method**Input:** A dataset D of network flows**Output:** The evaluation result (Recall, Precision, etc.)

```

1: preprocess( $D$ )
2:  $\varepsilon \leftarrow \text{findOptimalEpsilon}()$ 
3:  $\text{clusters} \leftarrow \text{DBSCAN}(\text{Dataset}, \varepsilon, \text{minPts})$ 
4:  $GD$ , global variable ▷ generated dataset
5: for each cluster  $C$  in  $\text{clusters}$  do
6:    $\text{row} = \text{analyzeCluster}(C)$ 
7:   emit  $\text{row}$  into  $GD$  and reset  $\text{row}$  value
8: end for
9:  $\text{model} = \text{loadClassifier}()$ 
10:  $\text{model.predict}(GD)$ 
11: assign cluster labels to the points inside them
12: reportEvaluationMetrics()

```

4.1 Phase 1: unsupervised learning

The first step of the proposed method is preprocessing the dataset that is done in line 1 of Algorithms 1 and 2. Preprocessing involves multiple procedures to prepare the dataset for the next major steps. The performed preprocessing procedures are described as follows:

- Removing high-correlated features: Among multiple features with more than 80% of correlation, only one feature is selected. We use the Pearson Correlation Coefficient, a metric with the resulting value in the range of $[-1, +1]$ that can indicate if there is a linear relationship between two variables (features). The mathematical equation of this metric for features f_i and f_j is shown in (1).

$$\text{corr}(f_i, f_j) = \frac{\sigma_{ij}}{\sigma_{f_i} \sigma_{f_j}} = \frac{E[(f_i - \mu_{f_i})(f_j - \mu_{f_j})]}{\sigma_{f_i} \sigma_{f_j}} \quad (1)$$

where σ_{ij} is the covariance of the two features, σ_{f_i} and σ_{f_j} are the standard deviation of features f_i and f_j , respectively, $E(x)$ represents the expected value of x , and μ_{f_i} is the average of f_i values.

- Removing the rows containing missing values
- Replacing the infinity values with a maximum possible value
- Normalizing the data using the min-max scaling method, i.e., rescaling the values of a feature to the range of [0,1]. If x is a value of feature f , x' is the mapped value for x is computed by:

$$x' = \frac{x - \min_f}{\max_f - \min_f} \quad (2)$$

where \max_f and \min_f are the maximum and minimum of f values. The mapped values computed by this equation are in the range of [0,1].

- Converting the categorical label column to a binary numerical one, such that *DDoS* \rightarrow 1 and *Benign* \rightarrow 0

After the preprocessing, DBSCAN is used for clustering the dataset, but initially, the optimal value for the ϵ parameter of DBSCAN is calculated in line 2. This operation is done using the method proposed by Elbata et al. [16], in which, at first, k lowest points' distances to all the points in the given dataset are sorted in ascending order. Then, they are plotted, and finally, the value where the drastic change occurs is regarded as the optimal ϵ . In other words, the elbow of the curve is the optimum value for ϵ . As they have reported, the value of k is determined to 3, and it does not significantly affect the calculated value of ϵ ; hence, we set $k = \text{minPts}$ in our method. Figure 5 shows an example illustrating how this method works, where the value of 0.15 at the elbow of the curve indicates that the optimal value of ϵ . Then, the optimal value for minPts is achieved by trial and error. Finally, the DBSCAN procedure is run in line 3 using the optimal ϵ and minPts , and the result of DBSCAN, which indicates the clustered data, is stored in variable *clusters*. The results show that the DDoS and benign traffic are adequately separated due to the clustering in phase 1.

4.2 Phase 2: analyzing clusters

During the second phase, we calculate some features for each cluster, as shown in function `analyzeCluster()` in Algorithm 3. Multiple statistical measures are selected as the features for the learning algorithms, which are illustrated in line 12 of the algorithm and also in the example of Fig. 4. In each cluster, the pairwise Euclidean distance between all pairs of the points² (inside the cluster) is calculated, and

² Each point here indicates a network flow in the raw dataset.

then the two points that have the maximum distance are considered for next steps. If $a = (f_{a1}, f_{a2}, \dots, f_{ak})$ and $b = (f_{b1}, f_{b2}, \dots, f_{bk})$ are two points such that f_k represents the k^{th} feature value³ of the points, the Euclidean distance is computed as follows:

$$\text{Euclidean distance}(a, b) = \sqrt{\sum_{i=1}^k (f_{ai} - f_{bi})^2} \quad (3)$$

Algorithm 3 The analyzeCluster() function

Input: A given Cluster C

Output: The calculated statistical measures for cluster C

```

1: function ANALYZECLUSTER(cluster  $C$ )
2:    $p_s, p_e \leftarrow \text{findTwoFurthestPoints}()$ 
3:    $r \leftarrow \text{distance}(p_s, p_e)$ 
4:    $np \leftarrow \text{number of partitions}$ 
5:    $row$ , local variable
6:   for  $i \leftarrow 0$  to  $np$  do
7:      $l \leftarrow i \cdot (1/np) \cdot r$ 
8:      $u \leftarrow (i + 1) \cdot (1/np) \cdot r$ 
9:      $points \leftarrow \{p \in C, l < \text{distance}(p, p_s) \leq u\}$ 
10:    if  $points$  is not empty then
11:       $pd \leftarrow \text{pairwiseDistance}(points)$ 
12:       $f \leftarrow \langle \min(pd), \max(pd), \text{mean}(pd), \text{std}(pd), |points|/|C| \rangle$ 
13:    else
14:       $f \leftarrow \langle -1, -1, -1, -1, 0 \rangle$ 
15:    end if
16:    append  $f$  into  $row$ 
17:  end for
18:  return  $row$ 
19: end function

```

The variables p_s and p_e in line 2 of the algorithm represent the two mentioned points (the furthest points in the cluster), and their distance is also stored in variable r in line 3. In lines 6-9, the distance r is divided to np disjoint partitions, and the points of each partition ($points$ in line 9) are processed in each run of the loop in line 6. The reason for the partitioning is detecting two patterns mentioned in Sect. 3 because, in this way, we can better discover how the points are distributed inside the cluster. The optimal value for the number of partitions (np defined in line 4) is selected by trial and error. Figure 4 also contains an example of the partitioning procedure for a cluster, where $np = 3$ (P_1, P_2 , and P_3). The notation of $|P_i|$ represents the number of points located in partition P_i ; for instance, $|P_3| = 11$ in Fig. 4 means there are 11 points located in partition 3 [of cluster C_1]. Note that C_i indicates cluster C_i , and $|C_i| = 30$ implies the number of points inside it. Subsequently, five features are calculated based on the pairwise distance of the points in each partition, including four statistical measures, and the fifth feature is the ratio of the points count in the

³ We used the features provided in the evaluation dataset, shown in Table 2 and described in Sect. 5.

Table 1 Structure of the GD, in which there are K clusters

Lines	Vectors										
1	min_1	max_1	μ_1	σ_1	$\frac{ P_1 }{ C_1 }$...	min_{np}	max_{np}	μ_{np}	σ_{np}	$\frac{ P_{np} }{ C_1 }$
2	min'_1	max'_1	μ'_1	σ'_1	$\frac{ P'_1 }{ C_2 }$...	min'_{np}	max'_{np}	μ'_{np}	σ'_{np}	$\frac{ P'_{np} }{ C_2 }$
...
K	min''_1	max''_1	μ''_1	σ''_1	$\frac{ P''_1 }{ C_k }$...	min''_{np}	max''_{np}	μ''_{np}	σ''_{np}	$\frac{ P''_{np} }{ C_k }$

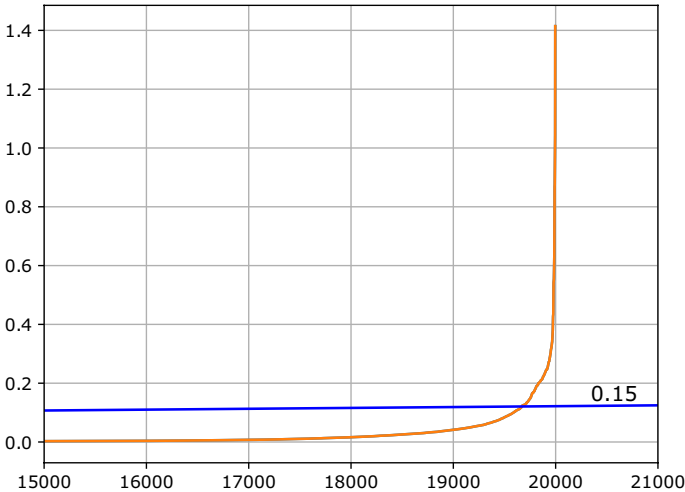


Fig. 5 An example for the optimal selection of ϵ ; $\epsilon = 0.15$

partition to the number of all points inside the cluster. The utilized statistical measures are *minimum*, *maximum*, *average*, and the *standard deviation* of the pairwise distance values. The features are calculated and assigned to the variable f in line 12, but if the current partition does not contain any points, the value of -1 is considered for the first four statistical measures and the value of 0 for the fifth feature in line 14. Since the mentioned statistical measures are positive for all non-empty partitions⁴, we consider the negative value of -1 for the empty partitions. Then, by line 16, f is appended to the variable row , formerly defined in line 5, and `analyzeCluster()` returns the computed features for the given cluster as variable row in line 18. When all the partitions are processed, there will be $5 \times np$ features computed and stored in row . Lastly, in the training step, the ratio of the number of DDoS-labeled points in the cluster to the number of all the points in it is regarded as the class label for the cluster and is appended to row by line 7 in Algorithm 1. In line 8 of Algorithm 1 and line 7 of Algorithm 2, row is appended to GD , which is defined in line

⁴ Due to the positiveness of the distance values; the value of the min, max, mean, and std of them are indeed positive.

Table 2 List of features utilized for evaluation

Feature	Description
Flow Duration	The time that a flow lasts
Min Packet Length	Minimum length of the packets in the flow
Total Fwd Packets	Number of all the packets in forward direction
FIN Flag Count	Number of packets with active FIN flag
RST Flag Count	Number of packets with active RST flag
URG Flag Count	Number of packets with active URG flag
PSH Flag Count	Number of packets with active PSH flag
ACK Flag Count	Number of packets with active ACK flag
CWE Flag Count	Number of packets with active CWE flag
Total Length of Fwd Packets	Size of all packets in forward direction
Fwd Packet Length Max	Maximum length of the packets in forward direction
Fwd Packet Length Min	Minimum length of the packets in forward direction
Bwd Packet Length Max	Maximum length of the packets in backward direction
Bwd Packet Length Min	Minimum length of the packets in backward direction
Flow Bytes/s	Number of Bytes transmitted in the flow per second
Down/Up Ratio	The ratio of download to upload
Flow Packets/s	Number of packets transmitted in the flow per second
Fwd Avg Bytes/Bulk	Average number of bytes per bulk in forward direction
Fwd Avg Packets/Bulk	Average number of packets per bulk in forward direction
Bwd Avg Packets/Bulk	Average number of packets per bulk in backward direction
Bwd Avg Bytes/Bulk	Average number of bytes per bulk in backward direction
Fwd Avg Bulk Rate	Average number of bulk rate in forward direction
Bwd Avg Bulk Rate	Average number of bulk rate in backward direction
Flow IAT Mean	Average time between two packets in the flow
Flow IAT Min	Minimum time between two packets in the flow
Fwd IAT Min	Minimum time between two packets in forward direction
Bwd IAT Total	Total time between two packets sent in backward direction
Bwd IAT Mean	Average time between two packets sent in backward direction
Fwd PSH Flags	Number of packets with active PSH flag in forward direction
Bwd PSH Flags	Number of packets with active PSH flag in backward direction
Fwd URG Flags	Number of packets with active URG flag in forward direction
Bwd URG Flags	Number of packets with active URG flag in backward direction
Init_Win_bytes_forward	All bytes sent in initial window in forward direction
Init_Win_bytes_backward	All bytes sent in initial window in backward direction
min_seg_size_forward	Minimum segment size observed in the forward direction
Active Mean	Average activation time of a flow before becoming idle
Active Std	Standard deviation of activation time of a flow before becoming idle
Idle Std	Standard deviation of idleness time of a flow before becoming active
Idle Min	Minimum idleness time of a flow before becoming active
Fwd Header Length	Total bytes of packet headers in forward direction
Fwd Packets/s	Number of forward packets per second
Bwd Packets/s	Number of backward packets per second

4. Table 1 illustrates how the GD and the utilized features look like. When all of the clusters are processed (lines 9 and 8 in Algorithms 1 and 2, respectively), the generated dataset GD will be completed. The DBSCAN runs in $O(n^2)$ [22], phase 2 of the proposed method runs in $O(|C| \times np)$ where $|C|$ is the number of the DBSCAN's output clusters, and np is the number of considered partitions, and the computational complexity of phase 3 varies according to the train or test procedures and also the utilized classification algorithm.

4.3 Phase 3: supervised learning

Having the GD , we build a classification model, after performing a preprocessing step. The class label is initially a continuous value, even though according to our observations, it is equal to 0 or 1 for most clusters due to high purity of them (99%), which indicates most of the clusters contain whether “DDoS” flows or “Benign” flows, only. The purity [33] of each round of a clustering algorithm is defined in (4):

$$purity = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j| \quad (4)$$

where N is the number of all points used in the clustering procedure, $\Omega = \{\omega_1, \dots, \omega_k\}$ is the set of resulted clusters, and $C = \{c_1, \dots, c_j\}$ is the set of the class labels. Then, multiple classification models on GD are trained saved in lines 10 and 11 of Algorithm 1. Loading the model and testing is performed in lines 9 and 10 of Algorithm 2. Note that the predicted label in this step are with respect to the clusters, not the network flows; hence, we assign the label of each cluster to all the flows inside it (line 11). Finally, the evaluation metrics are measured in line 12, which are described in Sect. 5.

5 Evaluation and results

We use the Python programming language for the implementation of the proposed method on the Linux operating system. The DBSCAN clustering is conducted using the Scikit-learn library [39], and all of the classification algorithms are performed by the MLib library [35] of Apache Spark framework with a cluster consists of 2 nodes with 4 CPU cores (8 cores in total). All the datasets' records are processed through a window with the size of 20,000 records (flows).

5.1 Datasets and attack types

We conduct the evaluation by training the proposed method on the CICIDS2017 dataset and testing on the CICDDoS2019 dataset. The CICIDS2017 is proposed by Sharafaldin et al. [49] and provides DDoS and benign traffic flows. The attack included in this dataset is produced by a tool called LOIC, which generates TCP, UDP, and HTTP flood attacks. The records of the dataset indicate network flows labeled to a binary value of [DDoS, Benign]. The CICDDoS2019 proposed by

Sharafaldin et al. [50] is captured from a completely different network but shares a similar feature set (80 features) to the CICIDS2017 and provides many additional DDoS attack types. Specifically, five different types of attack at CICDDoS2019 are as below:

- **SYN flood attack:** In this type of attack [9], the attacker sends a massive number of TCP SYN segments to the victim, which is usually conducted by the aim of a botnet. However, the attacker will not send the SYN-ACK packet after receiving the ACK from the victim; therefore, the victim's resources allocated to the half-open connection will be dissipated.
- **LDAP reflection-amplification attack:** Lightweight Directory Access Protocol (LDAP) is used to gather information from a server that provides a directory service. This protocol has been exploited by the attackers to fulfill the DDoS attack in such a manner as to send LDAP requests with the spoofed victim's IP address as the source IP address to many directory servers. Consequently, the servers send back their response to the victim and accordingly overwhelm it [4].
- **Portmap reflection-amplification attack:** Portmap (portmapper) is a service based on the ONC/RPC protocol, which provides calling a procedure in a program so that the procedure can be executed on another machine over the network [34]. By sending a request to a machine running the portmap service, it responds with the available programs that provide the ONC/RPC service and their corresponding port numbers. It enables attackers to abuse the portmap, spoofing the target's IP address and sending requests to many computer servers that provide the portmap service over the Internet. As a result, the target will be bombarded with the incoming portmap responses [30].
- **MSSQL reflection-amplification attack:** The attackers may utilize the Microsoft SQL Server (MSSQL) to perform the DDoS attacks by abusing the Microsoft SQL Server Resolution Protocol (MC-SQLR). Clients use MC-SQLR protocol to identify database instances in a database server or a cluster of database instances on a network; hence, the queried server will respond with a list of existing database instances. This protocol can also be exploited to launch a DDoS attack similar to the LDAP and Portmap reflection-amplification attacks [3].
- **NetBIOS reflection-amplification attack:** Network Basic Input/Output System (NetBIOS) is an API run over TCP/IP, enables computers to communicate and share their files on a local network, which is widely used by the Microsoft Windows computers in the same workgroup. For this purpose, a NetBIOS name must be set for the computers, with they identify each other. The NetBIOS Name Server (NBNS) is responsible for mapping the NetBIOS names of the computers to their network addresses. Thus, sending broadcast requests to a lot of NBNS's may result in receiving huge response traffic. This mechanism can be misused by attackers similar to the other reflection-amplification attacks so as to conduct DDoS attacks [6].

5.2 Evaluation of phase 1

After preprocessing, there are 42 features presented and described in Table 2 [19]. All the features in two mentioned datasets are produced using a tool called CICFlowMeter [18] by Habibi Lashkari et al. [15, 20].

To achieve a satisfactory efficacy for our proposed method, the initial clustering (the DBSCAN in phase 1) needs to be sufficiently accurate; therefore, we evaluate the purity in (4); a metric that indicates what extent the resulted clusters' points belong to the same class (Benign or DDoS) as well as the Homogeneity score [46], which shows how much these points belong to a single class. The homogeneity score formula is given as:

$$\text{homogeneity} = 1 - \frac{H(C|K)}{H(C)} \quad (5)$$

where $H(C|K)$ is the conditional entropy of the classes given the cluster assignments and is calculated by:

$$H(C|K) = - \sum_{c=1}^{|C|} \sum_{k=1}^{|K|} \frac{n_{c,k}}{n} \cdot \log \left(\frac{n_{c,k}}{n_k} \right) \quad (6)$$

and $H(C)$ is the entropy of the classes, calculated by:

$$H(C) = - \sum_{c=1}^{|C|} \frac{n_c}{n} \cdot p \log \left(\frac{n_c}{n} \right) \quad (7)$$

The notations used in (5), (6), and (7) are described as follows:

- K : The set of clusters
- C : The set of labels
- n : The number of total points in the given dataset
- n_c : The number of points belonging to class c
- n_k : The number of points belonging to cluster k
- $n_{c,k}$: The number of points belonging to class c which are assigned to cluster k

The example shown in Fig. 6 illustrates how these two metrics are calculated. In this example, there are two clusters and the number of all the data points is 5, including 3 points labeled with DDoS (positive) and 2 points with Benign (negative). The purity for this example is calculated as follows:

$$\text{purity} = \frac{\max(2, 1) + \max(1, 1)}{5} = 0.6$$

on the other hand, the homogeneity in this example is:

Fig. 6 An example for purity and homogeneity metrics

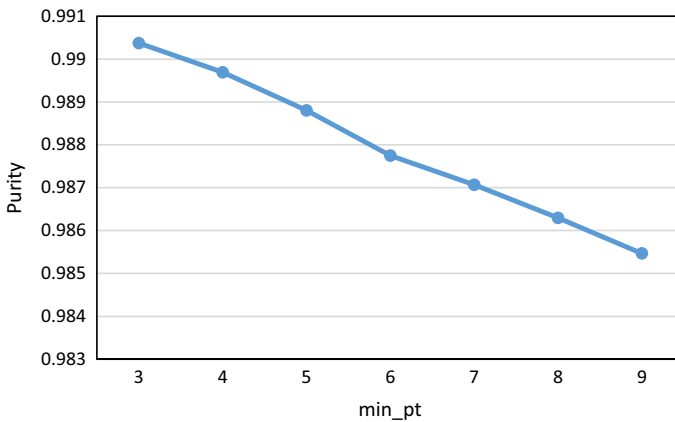
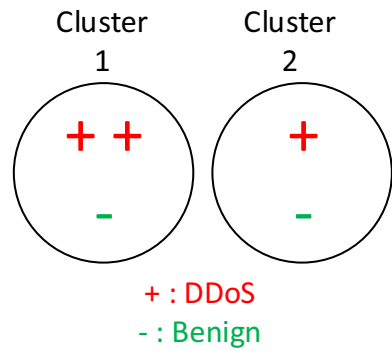


Fig. 7 Purity changes with *minPts*

$$\begin{aligned}
 & \textit{homogeneity} \\
 &= 1 - \frac{-\left(\frac{2}{5} \log\left(\frac{2}{3}\right) + \frac{1}{5} \log\left(\frac{1}{3}\right) + \frac{1}{5} \log\left(\frac{1}{2}\right) + \frac{1}{5} \log\left(\frac{1}{2}\right)\right)}{-\left(\frac{3}{5} \log\left(\frac{3}{5}\right) + \frac{2}{5} \log\left(\frac{2}{5}\right)\right)} \\
 &\approx 0.02
 \end{aligned}$$

We evaluate the resulted clusters of the DBSCAN on CICIDS2017, which shows 0.99 and 0.94 for the average purity and homogeneity of clusters, respectively, for *minPts* = 3 and average $\epsilon = 0.0227$. It demonstrates that in phase 1, the DDoS attack traffic and the benign one are sufficiently separated from each other with an accuracy of 99 %. As shown in Fig. 7, the purity decreases by increasing the parameter *minPts* with a linear slope. Moreover, Fig. 8 illustrates that the number of clusters sharply drops due to the rise in *minPts*. Increasing *minPts* forces the algorithm to make larger dense areas as clusters, which results in decreasing the number of clusters and increasing noises. Therefore, the purity also reduces.

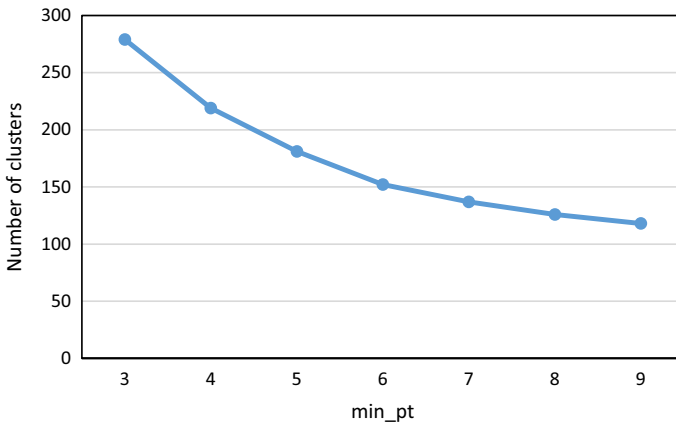


Fig. 8 Number of clusters changes with *minPts*

Table 3 Changes in the purity by the dataset sections count

	Number of split sections			
	5	10	15	20
Average purity	0.9818	0.9903	0.9906	0.9909

We also test the proposed method by applying different window lengths. In each time of evaluation, the dataset is split into a different number of sections, which are 5, 10, 15, and 20. By decreasing the number of applied sections (increasing length of the window), the purity of the clusters is slightly reduced, as shown in Table 3. Note that if we enhance the count of the sections, they will be smaller; consequently, fewer clusters noise will exist, and the purity increases.

There are some metrics and notations used for further evaluations, which are described as follows:

- True Positives (TP): It indicates the number of records belonging to the positive class, which the model correctly recognizes as “Positive.”
- True Negatives (TN): It indicates the number of records belonging to the negative class, which the model correctly recognizes as “Negative.”
- False Positives (FP): It indicates the number of records belonging to the negative class, which the model falsely recognizes as “Positive.”
- False Negatives (FN): It indicates the number of records belonging to the positive class, which the model falsely recognizes as “Negative.”

$$precision = \frac{TP}{TP + FP}$$

- Recall (also known as detection rate, sensitivity, and true positive rate): This metric indicates how many positive records are correctly recognized as positive by the model:

$$recall = \frac{TP}{TP + FN}$$

- False Positive Rate (FPR): This metric indicates how many negative records are falsely recognized as positive by the model:

$$FPR = \frac{FP}{FP + TN}$$

- Accuracy: It is a general criterion for measuring the overall ability of a model to make correct decisions and indicates how many negative records are falsely recognized as positive by the model:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- Positive likelihood ratio (LR+): This metric indicates how much the model succeeds in detecting positive samples while not identifying negative ones as Positive. LR+ is computed by the ratio of recall to FPR:

$$LR+ = \frac{recall}{FPR}$$

In this paper, “DDoS” flows are considered the positive class, and the “Benign” flows as the negative class.

5.3 Evaluation of phases 2 and 3

First, we evaluate multiple conventional ML methods that are DT, RF, NB, and SVM. As Fig. 9 illustrates, none of the mentioned algorithms was adequately able to detect the new attacks of the test dataset. The NB shows a value of zero for recall and precision, while the DT has the most recall (0.016), the RF the most LR+ (0.980), and the SVM the most FPR (0.148). The question is what the reasons behind these results are. The NB classifier assumes that if the feature f_i for a sample that belongs to class label C_i is equal to x , the probability of the sample [conditioned on C_i] will be obtained according to the normal distribution, meaning that:

$$P(f_i = x | C_i) = g(x, \mu_{C_i}, \sigma_{C_i}) \quad (8)$$

where μ_{C_i} and σ_{C_i} represent the mean and standard deviation, respectively, of feature f_i for the samples of class C_i , and $g(x, \mu, \sigma)$ indicates the Gaussian density function:

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right) \quad (9)$$

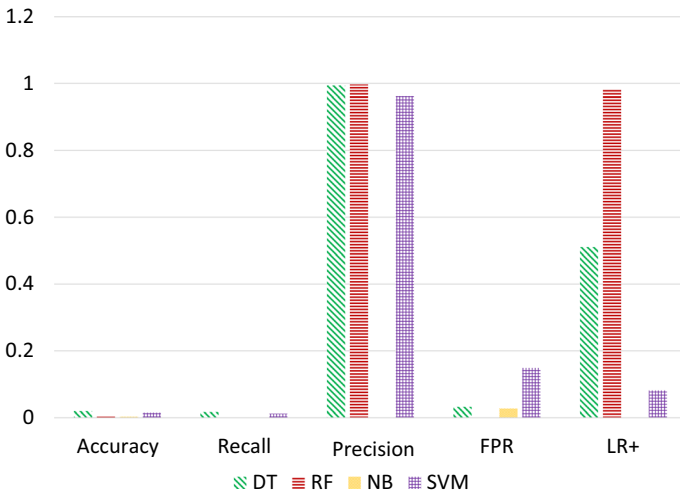


Fig. 9 Results of the conventional ML algorithms trained on CICIDS2017 and tested on CICDDoS2019

However, the results indicate that this assumption is not correct about the DDoS data; therefore, the NB classification results in $TP = 0$. On the other hand, the DT and RF not only do not make such an assumption but also suggest a higher ability to classify the data in a high feature space, and more importantly, they are likely to influence the worthwhile features more than others on the final decision by using metrics such as the Information Gain [40]. Moreover, the RF produces a lower FPR due to deciding upon the majority votes of multiple decision trees, suggesting that it cautiously decides.

Second, we consider DT, RF, NB, and SVM classification algorithms for phase 2 of the proposed method and evaluate them with values of 5, 10, 15, 20, and 25 for the np parameter. The achieved results are shown in Fig. 10.

The “X-based Proposed Method” notation used in the charts of this figure stands for the proposed method, which uses algorithm X in phase 2. According to the results shown in these charts, different algorithms and np values of phase 2 result in almost equal precision that is satisfactorily high, roughly 0.99. We observe that the FN is much higher than FP, suggesting that the proposed method rarely recognizes benign flows as DDoS, falsely. Thus, the precision is adequately high. On average, the *RF-based Proposed Method* has a lower recall than *DT-based Proposed Method* and *SVM-based Proposed Method* by about 39% and 65%; however, its FPR is also lower by roughly 71% and 67%, respectively. This result yet again leads us to the cautiousness of the RF classifier in decision-making. The *NB-based Proposed Method* does not provide sufficient effectiveness, presenting whether low recall and low FPR (≈ 0) or higher recall as well as higher FPR (≈ 0.4) due to the assumption about Gaussian distribution of the data in the NB classifier. The *SVM-based Proposed Method* (linear SVM) and *RF-based Proposed Method* are optimal on average in comparison with others due to higher recall while yielding a lower FPR, i.e., their LR+ are 1.8 and 1.7,

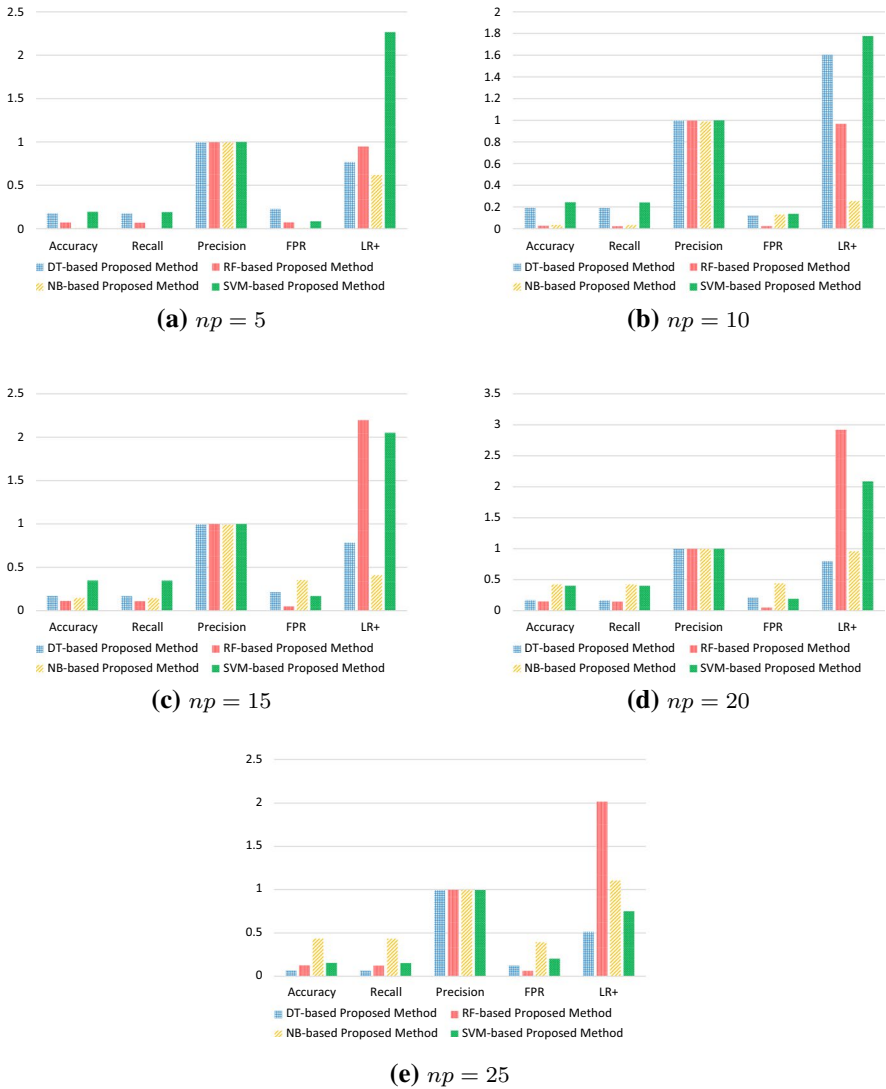


Fig. 10 Results of the proposed method trained on CICIDS2017 and tested on CICDDoS2019

respectively. However, it is equal to 0.86 and 0.79 for *DT-based Proposed Method* and *NB-based Proposed Method*, respectively. The result of the *SVM-based Proposed Method* suggests that the data after phases 1 and 2 are transformed in such a way that it can be better linearly classified. It should be noted that we believe a high value of recall, as well as a low value of FPR, are the main criteria for a DDoS detection algorithm; this is in view of the fact that recall shows the ability to detect malicious traffic and a high value of FPR indicates that the model practically recognizes the benign traffic as DDoS; therefore, it is indeed assisting the

attacker for reducing the availability in the network. Accordingly, LR+ plays a prominent role in evaluating algorithms.

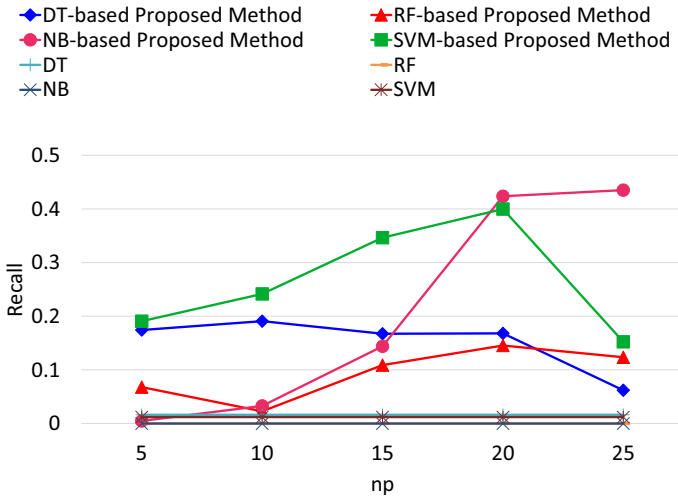
5.4 Comparison of the proposed method and existing approaches

Lastly, we compare the proposed method with the conventional ML algorithms concerning the recall and the FPR for multiple np values. As Fig. 11a shows, overall, the recall of the proposed method raises until $np = 20$, and then it is dropped, although it is higher than the recall of the conventional methods. In terms of the FPR shown in Fig. 11b, the *NB-based Proposed Method* significantly grows by increasing np . Yet again, this assumption that the features are related to the class label according to the normal distribution is the main reason for that. Others do not increase dramatically; *SVM-based Proposed Method* raises with a slight slope, while *RF-based Proposed Method* and *DT-based Proposed Method* are more robust because they can effectively take advantage of a large number of features. Among all cases of the proposed method, *RF-based Proposed Method* has the lowest FPR and is almost constant in different values of np . These results indicate that both over-partitioning and under-partitioning over clusters lead to negative effects, and $np = 20$ is the optimal number for this parameter.

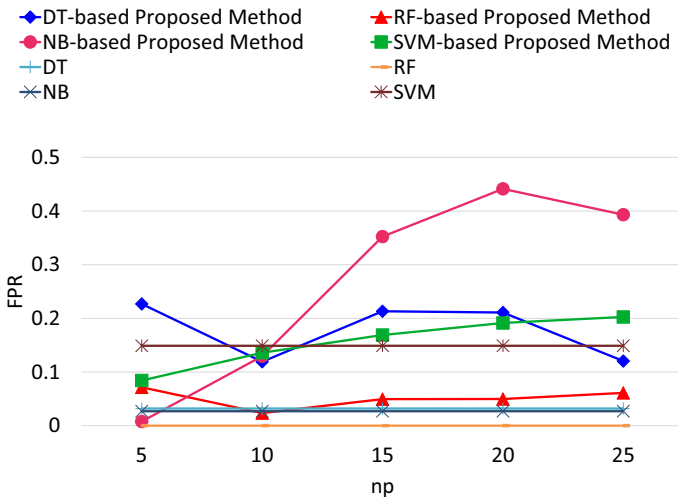
Many recent and related research studies propose conventional ML algorithms or hybrid and innovative methods for detecting DDoS attacks but evaluate them using the same dataset for both training and testing [8, 12, 13, 23–25, 31, 37, 41, 44, 45, 48, 56, 60]. The overall comparison between the proposed method and the ML algorithms used in the mentioned papers is presented in Table 4. The most optimal cases of the proposed method are when either the SVM is used in phase 2 or RF with $np = 20$. On average, *SVM-based Proposed Method* is the most optimal with regard to recall, accuracy, and FPR; however, the result of the RF is satisfactory when the number of features would be optimally high, i.e., $np = 20$. The highest LR+ is achieved by *RF-based Proposed Method_20*⁵ (2.92056), which is approximately three times higher than the highest LR+ of the conventional algorithms (RF). Furthermore, *SVM-based Proposed Method_20* provides roughly 0.40 for accuracy and recall, while their maximum value in conventional algorithms is approximately 0.02, achieved by DT. The results show that the accuracy and recall of the proposed method are improved by average 25.5 and 36.3 times compared to conventional machine learning methods, respectively. An important issue to note is that these massive improvements in our method to the conventional algorithms are not far-fetched. Since, the learning algorithms are extremely dependent on the data, and it is likely that the attack type, attack-conducting tools, network architecture and characteristics, as well as physical parameters such as noise and packet loss may deeply affect their final results.

Some related proposed methods are evaluated using different datasets. For example, Yang et al. [57] conducted an experiment, where the recall is optimized by 61

⁵ The proposed method that uses RF in phase 2 and $np = 20$.



(a) Recall



(b) FPR

Fig. 11 Comparison of recall in the conventional ML algorithms and the proposed methods by different values of np

times. However, by exchanging the test and train datasets but keeping the same Benign samples, it is about 0.34%. The method proposed by Liang and Znati [32] shows an improvement over conventional ML methods, but the train and test datasets are both the subsets of CICIDS2017, and the same attack and target networks are used for them. Moreover, the value of FPR, which is so important to evaluate

Table 4 Proposed method result in comparison with the conventional ML algorithms

Method and Ref.*	Accuracy	Recall	Precision	FPR	LR+
Decision Tree [37, 48]	0.01936	0.01634	0.99379	0.03201	0.51027
Random Forest [24, 45]	0.00322	0.00004	0.99676	0.00004	0.98054
Naïve Bayes [56, 60]	0.00309	0.00000	0.00000	0.02699	0.00000
SVM [14, 12]	0.01471	0.01204	0.96207	0.14893	0.08087
Average	0.01010	0.00711	0.73816	0.05199	0.39292
RF-based Proposed Method_20**	0.14792	0.14542	0.99891	0.04979	2.92056
SVM-based Proposed Method_20**	0.40100	0.39970	0.99848	0.19143	2.08804
SVM-based Proposed Method_15**	0.34789	0.34634	0.99844	0.16884	2.05132
SVM-based Proposed Method_5**	0.19312	0.19081	0.99859	0.08422	2.26567
Average	0.26726	0.26533	0.99859	0.12198	2.31444

* The conventional ML method and the related papers that used the corresponding ML method to detect DDoS attacks

** “X-based Proposed Method_Y” indicates a case of the proposed method which uses algorithm “X” in phase 2, and $np = Y$

a DDoS detection system, is not reported. The comparison between this paper and some of the existing approaches is shown in Table 5.

6 Conclusion

In this paper, we propose a hybrid machine-learning-based method for detecting unprecedented DDoS attacks. The proposed method works in three phases. It initially utilizes the DBSCAN algorithm to cluster and separate the DDoS network flows from the benign flows (unsupervised phase). Then, it partitions each cluster and analyzes it using some statistical measures (cluster analyzing). From the calculated measures, the classification step (supervised phase) determines whether the clusters contain DDoS flows or benign traffic. The result shows that the clusters are sufficiently pure regarding the class label (DDoS or Benign) they contain. The training dataset is the CICIDS2017 (DDoS subset), and the one used for testing is the CICDDoS2019. We utilize the DBSCAN algorithm for the unsupervised phase and some of the widely used classification algorithms in the supervised phase. Moreover, we analyze the clusters by partitioning the data points inside them. The evaluation result demonstrates that considering the properly tuned proposed method (*RF-based Proposed Method_20*), our method is about 198% more effective (from the LR+ point of view) in detecting the unprecedented types of DDoS attack in comparison with the best conventional ML method (Random Forest) used in recent related work. The average improvement for the recall metric of the tuned proposed method over the classification algorithms used in the literature review is approximately 36.3%.

Table 5 Comparison between the proposed method and the related work

Paper	Method	Unknown test data ¹		Data date ²	Big data ³	Functionality
		Attack	Network			
Idhammad et al. 2018 [25]	Co-clustering + classification	No	No	2015	No	Entropy calculation and Co-clustering the flows and then classification using Extra-trees
Roopak et al. 2019 [45]	Classification	No	No	2017	No	Classifying data using ANN and conventional ML algorithms
Saimi et al. 2020 [48]	Classification	No	No	2016	No	Conducting classification methods on data to detect five attack types
Li et al. 2020 [31]	Entropy analysis + recurrent ANN	No	No	2019 and a synthetic dataset	No	Calculating a joint entropy of source IP and destination IP addresses and using LSTM for eliminating noise effect
Liang and Znati 2019 [32]	Supervised ANN	Yes	No	2017	No	Feeding a stream of the packets into an LSTM network to detect the attack
Yang et al. 2020 [57]	Unsupervised ANN	Yes	No	2017 (and a synthetic dataset)	No	Using an Autoencoder network to calculate the reconstruction error of the incoming packets to detect the attack
This paper	Clustering + Classification	Yes	Yes	2019	Yes	Density-based clustering and then analyzing cluster points distribution using statistical measures and then classification

¹ Is evaluation conducted on different data (attack or network) than the train data? ² The year of the most up-to-date dataset used in the paper ³ Is a big data framework on a cluster of computers being used for processing?

Funding The authors did not receive any financial support for this study.

Data Availability The data used in the paper will be available upon request.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Code availability The code will be available after obtaining permission from the Yazd University.

References

1. CAIDA UCSD Network Telescope Traffic Samples (formerly the Backscatter Dataset). https://www.caida.org/data/passive/backscatter_dataset.xml. Accessed: Jan. 23, 2020
2. Mininet: An Instant Virtual Network on your Laptop (or other PC) - Mininet. <http://mininet.org/>. Accessed: May 23, 2021
3. MS SQL PacketResolution DoS: Attack signature - symantec corp. https://www.symantec.com/security_response/attacksignatures/detail.jsp?asid=20533. Accessed: Jan. 7, 2021
4. Old Protocols, New Exploits: LDAP Unwittingly Serves DDoS Amplification Attacks. <https://www.f5.com/labs/articles/threat-intelligence/old-protocols-new-exploits-ldap-unwittingly-serves-ddos-amplification-attacks-22609>. Accessed: Jan. 20, 2021
5. Abubakar R, Aldegheishem A, Majeed MF, Mehmood A, Maryam H, Alrajeh NA, Maple C, Jawad M (2020) An effective mechanism to mitigate real-time DDoS attack. *IEEE Access* 8:126215–126227. <https://doi.org/10.1109/ACCESS.2020.2995820>
6. Akamai Technologies (2021) Inc.: Threat advisory: Netbios name server, rpc portmap and sentinel reflection DDoS. <https://www.akamai.com/uk/en/multimedia/documents/state-of-the-internet/ddos-reflection-netbios-name-server-rpc-portmap-sentinel-udp-threat-advisory.pdf>. Accessed: Jan. 31, 2021
7. Bakker JN, Ng B, Seah WKG (2018) Can machine learning techniques be effectively used in real networks against DDoS attacks? In: 2018 27th International Conference on Computer Communication and Networks (ICCCN), pp. 1–6. Hangzhou, China (2018). <https://doi.org/10.1109/ICCCN.2018.8487445>
8. Balkanlı E, Alves J, Zincir-Heywood AN (2014) Supervised learning to detect DDoS attacks. In: 2014 IEEE Symposium on Computational Intelligence in Cyber Security (CICS), pp. 1–8. Orlando, FL, USA (2014). <https://doi.org/10.1109/CICYBS.2014.7013367>
9. Bogdanoski M, Suminoski T, Risteski A (2013) Analysis of the syn flood dos attack. *Int J Comput Netw Inform Secur (IJCNIS)* 5(8):1–11. <https://doi.org/10.5815/ijcnis.2013.08.01>
10. University of California, I. (2021) KDD Cup 1999 Data. <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>. Accessed: May 23, 2021
11. Cambiasoa E, Chiola G, Aiello M (2019) Introducing the slowdrop attack. *Comput Netw* 150:234–249. <https://doi.org/10.1016/j.comnet.2019.01.007>
12. Chen J, Yang Yt, Hu Kk, Zheng Hb, Wang Z (2019) DAD-MCNN: DDoS attack detection via multi-channel CNN. In: Proceedings of the 2019 11th International Conference on Machine Learning and Computing, ICMLC '19, p. 484–488. Association for Computing Machinery, Zhuhai, China (2019). <https://doi.org/10.1145/3318299.3318329>
13. Deepa V, Sudar KM, Deepalakshmi P (2018) Detection of DDoS attack on SDN control plane using hybrid machine learning techniques. In: 2018 International Conference on Smart Systems and Inventive Technology (ICSSIT), pp. 299–303. IEEE, Tirunelveli, India (2018). <https://doi.org/10.1109/ICSSIT.2018.8748836>

14. Doshi R, Apthorpe N, Feamster N (2018) Machine learning DDoS detection for consumer internet of things devices. In: 2018 IEEE Security and Privacy Workshops (SPW), pp. 29–35. San Francisco, CA, USA (2018). <https://doi.org/10.1109/SPW.2018.00013>
15. Draper-Gil G, Lashkari AH, Mamun MSI, A Ghorbani A (2016) Characterization of encrypted and vpn traffic using time-related features. In: Proceedings of the 2nd International Conference on Information Systems Security and Privacy - ICISSP., pp. 407–414. INSTICC, SciTePress (2016). <https://doi.org/10.5220/0005740704070414>
16. Elbatta MT, Ashour WM (2013) A dynamic method for discovering density varied clusters. *Int J Signal Process Image Process Pattern Recognit* 6(1):123–134
17. Ester M, Kriegel HP, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96, p. 226–231. AAAI Press (1996)
18. Habibi Lashkari A, Draper Gil G, Mamun MSI (2021) Github - ahlashkari/cicflowmeter. <https://github.com/ahlashkari/CICFlowMeter>. Accessed on: Jun. 2, 2021
19. Habibi Lashkari A, Draper Gil G, Mamun MSI (2021) Github - ahlashkari/cicflowmeter/readme. <https://github.com/ahlashkari/CICFlowMeter/blob/master/ReadMe.txt>. Accessed on: Jun. 2, 2021
20. Habibi Lashkari A, Draper Gil G, Mamun MSI, Ghorbani AA (2017) Characterization of tor traffic using time based features. In: Proceedings of the 3rd International Conference on Information Systems Security and Privacy - ICISSP., pp. 253–262. INSTICC, SciTePress (2017). <https://doi.org/10.5220/0006105602530262>
21. Haider S, Akhunzada A, Mustafa I, Patel TB, Fernandez A, Choo KKR, Iqbal J (2020) A deep cnn ensemble framework for efficient ddos attack detection in software defined networks. *IEEE Access* 8:53972–53983. <https://doi.org/10.1109/ACCESS.2020.2976908>
22. Han J, Pei J, Kamber M (2012) Data mining: concepts and techniques, third edn. Elsevier (2012). <https://doi.org/10.1016/C2009-0-61819-5>
23. Hosseini S, Azizi M (2019) The hybrid technique for ddos detection with supervised learning algorithms. *Comput Netw* 158:35–45. <https://doi.org/10.1016/j.comnet.2019.04.027>
24. Hou J, Fu P, Cao Z, Xu A (2018) Machine learning based DDoS detection through netflow analysis. In: MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM), pp. 1–6. Los Angeles, CA, USA (2018). <https://doi.org/10.1109/MILCOM.2018.8599738>
25. Idhammad M, Afdel K, Belouch M (2018) Semi-supervised machine learning approach for DDoS detection. *Appl Intell* 48(10):3193–3208. <https://doi.org/10.1007/s10489-018-1141-2>
26. Imperva Research Labs (2021) DDoS Attacks in the Time of COVID-19. https://www.imperva.com/resources/reports/Imperva_DDoSAttacksCOVID-19_Report_20201217.pdf. Accessed: Mar. 3, 2021
27. Kalathiripi R (2021) Regression coefficients of traffic flow metrics (rctfm) for ddos defense in iot networks. *Int J Commun Syst* 34(6):e4330. <https://doi.org/10.1002/dac.4330>
28. Kupreev O, Badovskaya E, Gutnikov A (2021) DDoS attacks in q4 2019. <https://securelist.com/ddos-report-q4-2019/96154/>. Accessed: Mar. 3, 2021
29. Kupreev O, Badovskaya E, Gutnikov A (2021) DDoS attacks in q4 2020. <https://securelist.com/ddos-attacks-in-q4-2020/100650/>. Accessed: Mar. 3, 2021
30. Labs BL (2021) A new DDoS reflection attack: Portmapper; an early warning to the industry. <https://blog.lumen.com/a-new-ddos-reflection-attack-portmapper-an-early-warning-to-the-industry/>. Accessed on: May. 20, 2021
31. Li J, Liu M, Xue Z, Fan X, He X (2020) RTVD: a real-time volumetric detection scheme for ddos in the internet of things. *IEEE Access* 8:36191–36201
32. Liang X, Znati T (2019) A long short-term memory enabled framework for ddos detection. In: 2019 IEEE Global Communications Conference (GLOBECOM), pp. 1–6. Waikoloa, HI, USA (2019). <https://doi.org/10.1109/GLOBECOM38437.2019.9013450>
33. Manning CD, Schütze H, Raghavan P (2008) Introduction to information retrieval. Cambridge University Press, Cambridge. <https://doi.org/10.1017/CBO9780511809071>
34. Marshall AD (2005) Remote procedure calls (rpc). In: Programming in C - UNIX System Calls and Subroutines using C. Cardiff School of Computer Science & Informatics (2005). <https://users.cs.cf.ac.uk/Dave.Marshall/C/node33.html>
35. Meng X, Bradley J, Yavuz B, Sparks E, Venkataraman S, Liu D, Freeman J, Tsai D, Amde M, Owen S, Xin D, Xin R, Franklin MJ, Zadeh R, Zaharia M, Talwalkar A (2016) Mllib: machine learning in apache spark. *J Mach Learn Res* 17(1):1235–1241

36. Mirsky Y, Doitshman T, Elovici Y, Shabtai A (2018) Kitsune: an ensemble of autoencoders for online network intrusion detection. arXiv preprint [arXiv:1802.09089](https://arxiv.org/abs/1802.09089) (2018). <https://doi.org/10.14722/ndss.2018.23204>
37. Morfino V, Rampone S (2020) Towards near-real-time intrusion detection for iot devices using supervised learning and apache spark. *Electronics* 9(3):444. <https://doi.org/10.3390/electronics9030444>
38. Moustafa N, Slay J (2015) Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: 2015 military communications and information systems conference (MilCIS), pp. 1–6. IEEE, Canberra, ACT, Australia (2015). <https://doi.org/10.1109/MilCIS.2015.7348942>
39. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in python. *J Mach Learn Res* 12:2825–2830
40. Quinlan JR (1986) Induction of decision trees. *Mach Learn* 1(1):81–106. <https://doi.org/10.1007/BF00116251>
41. Rahman O, Quraishi MAG, Lung C (2019) DDoS attacks detection and mitigation in SDN using machine learning. In: 2019 IEEE World Congress on Services (SERVICES), 2642–939X, pp. 184–189. Milan, Italy (2019). <https://doi.org/10.1109/SERVICES.2019.000051>
42. Rezapour A, Tzeng WG (2021) RI-shield: mitigating target link-flooding attacks using sdn and deep reinforcement learning routing algorithm. *IEEE Transactions on Dependable and Secure Computing* pp. 1–1 (2021). <https://doi.org/10.1109/TDSC.2021.3118081>
43. Rios VdM, Inácio PR, Magoni D, Freire MM (2021) Detection of reduction-of-quality DDoS attacks using fuzzy logic and machine learning algorithms. *Computer Networks* **186**, 107792 (2021). <https://doi.org/10.1016/j.comnet.2020.107792>
44. Roempluk T, Surinta O (2019) A machine learning approach for detecting distributed denial of service attacks. In: 2019 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT-NCON), pp. 146–149. Nan, Thailand (2019). <https://doi.org/10.1109/ECTI-NCON.2019.8692243>
45. Roopak M, Tian GY, Chambers J (2019) Deep learning models for cyber security in iot networks. In: 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), pp. 0452–0457. IEEE, Las Vegas, NV, USA (2019). <https://doi.org/10.1109/CCWC.2019.8666588>
46. Rosenberg A, Hirschberg J (2007) V-measure: A conditional entropy-based external cluster evaluation measure. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pp. 410–420. Association for Computational Linguistics, Prague, Czech Republic (2007). <https://www.aclweb.org/anthology/D07-1043>
47. Rossow C (2014) Amplification hell: Revisiting network protocols for DDoS abuse. In: The Netw. and Distrib. Syst. Secur. Symp. (2014). <https://doi.org/10.14722/ndss.2014.23233>
48. Saini PS, Behal S, Bhatia S (2020) Detection of DDoS attacks using machine learning algorithms. In: 2020 7th International Conference on Computing for Sustainable Global Development (INDIACom), pp. 16–21. IEEE, New Delhi, India (2020). <https://doi.org/10.23919/INDIACom49435.2020.9083716>
49. Sharafaldin I, Habibi Lashkari A, Ghorbani AA (2018) Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy, pp. 108–116. Funchal, Madeira, Portugal (2018). <https://doi.org/10.5220/0006639801080116>
50. Sharafaldin I, Lashkari AH, Hakak S, Ghorbani AA (2019) Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In: 2019 International Carnahan Conference on Security Technology (ICCST), pp. 1–8. Chennai, India (2019). <https://doi.org/10.1109/CCST.2019.8888419>
51. Shieh CS, Lin WW, Nguyen TT, Chen CH, Horng MF, Miu D (2021) Detection of unknown ddos attacks with deep learning and gaussian mixture model. *Appl Sci*. <https://doi.org/10.3390/app11115213>
52. Shiravi A, Shiravi H, Tavallaee M, Ghorbani AA (2012) Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Comput Secur* 31(3):357–374. <https://doi.org/10.1016/j.cose.2011.12.012>

53. Tavallaee M, Bagheri E, Lu W, Ghorbani AA (2009) A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6. Ottawa, ON, Canada (2009). <https://doi.org/10.1109/CISDA.2009.5356528>
54. Ujjan RMA, Pervez Z, Dahal K, Bashir AK, Mumtaz R, González J (2020) Towards sFlow and adaptive polling sampling for deep learning based DDoS detection in SDN. *Futur Gener Comput Syst* 111:763–779. <https://doi.org/10.1016/j.future.2019.10.015>
55. Wang M, Lu Y, Qin J (2020) A dynamic MLP-based DDoS attack detection method using feature selection and feedback. *Comput Secur.* <https://doi.org/10.1016/j.cose.2019.101645>
56. Wani AR, Rana QP, Saxena U, Pandey N (2019) Analysis and detection of DDoS attacks on cloud computing environment using machine learning techniques. In: 2019 Amity International Conference on Artificial Intelligence (AICAI), pp. 870–875. Dubai, United Arab Emirates (2019). <https://doi.org/10.1109/AICAI.2019.8701238>
57. Yang K, Zhang J, Xu Y, Chao J (2020) Ddos attacks detection with autoencoder. In: NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium, pp. 1–9. Budapest, Hungary (2020). <https://doi.org/10.1109/NOMS47738.2020.9110372>
58. Yungaicela-Naula NM, Vargas-Rosales C, Perez-Diaz JA (2021) SDN-based architecture for transport and application layer DDoS attack detection by using machine and deep learning. *IEEE Access* 9:108495–108512. <https://doi.org/10.1109/ACCESS.2021.3101650>
59. Zaharia M, Xin RS, Wendell P, Das T, Armbrust M, Dave A, Meng X, Rosen J, Venkataraman S, Franklin MJ, Ghodsi A, Gonzalez J, Shenker S, Stoica I (2016) Apache spark: a unified engine for big data processing. *Commun ACM* 59(11):56–65. <https://doi.org/10.1145/2934664>
60. Zhou B, Li J, Wu J, Guo S, Gu Y, Li Z (2018) Machine-learning-based online distributed denial-of-service attack detection using spark streaming. In: 2018 IEEE International Conference on Communications (ICC), pp. 1–6. Kansas City, MO, USA (2018). <https://doi.org/10.1109/ICC.2018.8422327>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.