Computer Networks 164 (2019) 106906

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

# On the performance of intelligent techniques for intensive and stealthy DDos detection

Xiaoyu Liang*, Taieb Znati

*Department of Computer Science, University of Pittsburgh, USA*

ARTICLE INFO

*Article history:*
Received 8 March 2019
Revised 23 August 2019
Accepted 16 September 2019
Available online 18 September 2019

*Keywords:*
DDoS detection review
Machine learning
Empirical analysis

ABSTRACT

Distributed Denial of Services (DDoS) attacks continue to be one of the most challenging threats to the Internet. The intensity and frequency of these attacks are increasing at an alarming rate. With the promising results presented by Machine Learning (ML) techniques in variety fields, researchers have proposed numerous intelligent schemes to defend against DDoS attacks and mitigate their impact. This paper presents a taxonomy of the ML-based DDoS detection schemes, focusing on the important features and mechanisms that each scheme uses to detect and mitigate the impact of these attacks. The taxonomy is developed based on a thorough and extensive review of the literature, focusing on the most prominent and highly cited schemes that have been proposed over the last decade. The taxonomy is then used as a basis for the development of a framework to conduct a comprehensive empirical evaluation of the basic mechanisms underling the design of the selected ML-based DDoS defense schemes against a variety of attack scenarios. Rather than dealing with the specific details of a particular DDoS defense scheme, this work focuses on the "building blocks" of the intelligent DDoS detection and prevention schemes. The intelligent mechanisms underlying the selected schemes are implemented and evaluated using different performance metrics. The impact of different influential factors are also explored, including the observable traffic proportions, attack intensities and the "Class Imbalance Problem" of ML-based DDoS detection. The results of the comparative analysis show that no single technique outperforms all others in all test cases. Furthermore, the results underscore the need for a method oriented feature selection model to enhance the capabilities of ML-based detection techniques. Finally, the results show that the class imbalance problem significantly impact performance, underscoring the need for further research to address this problem and ensure high-quality DDoS detection in real-time.



## 1. Introduction

With the advent of new computing paradigms, such as Cloud and Mobile computing, and the emergence of pervasive technology, such as the Internet of Things, Distributed Denial of Services (DDoS) attacks have been growing dramatically in volume, frequency, sophistication and impact, making it one of the most challenging threats in the Internet [1]. On March 2015, GitHub faced a massive DDoS attack. The attack lasted for almost one week and caused significant damages [2]. On October 2016, a series of massive DDoS attacks were perpetrated against Dyn's DNS servers, causing the disruption of multiple major websites, including Airbnb, Netflix, and Spotify [3]. On September 2017, UK National Lottery was targeted by a DDoS attack during peak time operations, forcing the organization to take offline its lottery website and mobile App for about 90 minutes [4]. On February 2018, Github was taken offline for about 10 minutes by a DDoS attack, which peaked at 1.35 Tbps [5]. These recent attack events underscore the increasing intensity of these attacks and demonstrate the severe impact they continue to inflict on critical infrastructure.

A DDoS attack is an attempt to disrupt legitimate access to targeted computing resources, including computer systems, network devices, servers and web applications. DDoS attacks are typically launched from a very large number of distributed, remotely controlled devices, organized into botnets and aimed at attacking the same target [6]. It is often the case that a significant number of these devices are unwitting members of botnets, operating with spoofed network addresses, making it hard to identify attack sources. At one extreme, a DDoS attack manifests itself in the form of "complete" denial of service access to legitimate users. Attacks in this category are designed to deplete computing resources, by sending simultaneously an excessive number of computing requests, thereby flooding the network resources intensively with malicious traffic. The objective is to render the target completely

* Corresponding author.
*E-mail addresses:* xil160@pitt.edu (X. Liang), znati@cs.pitt.edu (T. Znati).

https://doi.org/10.1016/j.comnet.2019.106906
1389-1286/

unresponsive to legitimate users. While these traditional DDoS attacks remain a menace, a new form of DDoS attacks has emerged. These new attacks no longer aim to completely bring down the server. Instead, their goal is to frequently cause intermediate service slowness and intolerable network delays. Despite significant advances in the state-of-the-art of system and network security, defending against DDoS attacks remains a challenging problem.

Numerous schemes have been proposed to mitigate the impact of DDoS attacks. Several surveys provide an extensive classifications on both DDoS attacks and defense mechanisms, focusing on the advantages and limitations of the proposed schemes [7–14]. These schemes can be categorized into two main classes: rule-based and anomaly-based DDoS defense schemes. Rule-based defense schemes, such as Snort [15], are very effective in detecting "known" attacks, whereby a "signature", which captures crucial features of the attack, is available in advance. However, these strategies require "manually" configuring DDoS detection rules, based on a deep understanding of the attack traffic characteristics and behavior. The need of prior knowledge to characterize attack traffic, coupled with the need of manual intervention, may limit the scope of our study and hinder our ability to assess the full capabilities of evaluated schemes. Hence, we exclude rule-based defense schemes from our work, and mainly focus on anomaly-based DDoS defense schemes.

Schemes evaluated in this work are capable to "automatically" learn traffic behavior and characteristics, and detect potential DDoS attacks, while minimizing false alarms. The early schemes use a number of approaches to defend DDoS attacks, including stochastic analysis to monitor network traffic flows' behavior and exploit the entropy of network traffic to identify normal behavior and detect anomalous intrusion events.

Recently, the trend to mitigate the impact of DDoS attacks is to incorporate "intelligence" into the defense architecture, leveraging machine learning techniques to classify and detect malicious traffic. A number of research surveys have been proposed to assess, analyze and compare the performance of these intelligent schemes [12,16–18]. Although informative of the state-of-the-art in intelligent DDoS detection and prevention, the intricacies and inherent technical implementation details of the proposed schemes stand in the way of a fair comparative assessment of the studied schemes, thereby affecting the outcome of the analysis. Furthermore, the focus on specific schemes further constrain the applicability of the results to other schemes. As such, the outcome of these studies remain inconclusive. Very little effort has been dedicated to "holistically" evaluate the performance of intelligent solutions to DDoS defense. This stems from the fact that such an undertaking is not realistic, given the large number of schemes proposed in the literature. A closer look at the proposed schemes, however, reveals that at the core of their design is a set of AI-centered "building blocks". Rather than carrying out an exhaustive comparative analysis of specific schemes, this paper takes a different and unique approach to investigate the performance of the main building blocks frequently used in intelligent DDoS attack detection and prevention.

To achieve the above objective, a literature survey was carried out to identify the most prominent intelligent DDoS detection schemes in a vast DDoS solution space, understand their similarities and differences and assess their effectiveness in an expansive and fair manner. To this end, we set a number of criteria to identify a core set of "representative" class of intelligent strategies and build a taxonomy that provides a global view of the DDoS problem and enables a comprehensive description of the solution space in this field of research. The selection criteria to include how essential is the use intelligence to the design of the proposed scheme, the publication date of the paper describing the work, and the number of citations as reported by Google Scholar. The availability of a software implementation of the proposed scheme, although not required, was considered favorably in the selection process, especially for similar schemes.

Based on the outcome of the survey, a framework was developed to assess the performance of the selected schemes and carry out an empirical comparative analysis of these schemes in a heterogeneous environment, under different attack scenarios. The "building blocks" of all selected schemes were fully implemented in this study, focusing on capturing the important features of each scheme. The empirical analysis attempts to answer the question of which technique, if any, outperforms all others. To this end, we empirically investigate the factors that influence the performance of these techniques, including different network protocols, feature types, observable network traffic proportions, attack traffic intensities and the class imbalanced problem. We then carry out a sensitivity analysis of the schemes to these factors. The aim is to gain better understanding of how a given factor influences the performance of each technique. The main novelty and contributions of the paper are as follows:

- The development of a taxonomy, based on a thorough review of intelligent DDoS detection schemes, which captures the current trend in ML-based DDoS detection. Guided by a rigorous selection process, (i) the most prominent and highly cited schemes were identified, and (ii) a core set of "building blocks" consistently used in intelligent DDoS detection and prevention strategies was derived.
- A comprehensive classification and performance analysis approach, focusing on the building blocks of intelligent detection strategies, as opposed to the "intricacies" of a particular scheme, used to counter the DDoS problem. The objective is to carry out a comparative analysis study that can be used to highlight commonalities and differences among DDoS detection strategies, while preserving their important features and behavior. The study will enable researchers to gain a better understanding of the strength and limitations of a multitude of solutions in a vast DDoS detection problem space.
- The development of a comprehensive performance evaluation and comparative analysis framework of the selected intelligent strategies, using a collection of real-world network traffic data and their associated meta-data. The data is organized into different dataset benchmarks, tailored for different test scenarios and experimental evaluation studies. A set of evaluation metrics, namely *accuracy, sensitivity* and *specificity*, are used to assess the performance and carry out a comparative analysis of these schemes.
- The development of a rich set of attack scenarios, using a combination of attack and legitimate traffic data from different sources. The scenarios are engineered to exercise a wide spectrum of DDoS attack features and characteristics.
- The design of a series of experiments, using different attack scenarios, to explore the impact of packet- and flow-level features, and the observable traffic proportions on the performance of the selected techniques. A specific focus of these experiments is on the impact of the common "Class Imbalance Problem", in terms of the ratio between legitimate and attack traffic, on the performance of these intelligent techniques. In practice, this problem is common in various disciplines, including anomaly detection. To the best of our knowledge, this is the first empirical study that addresses the Class Imbalance Problem impact on the performance of DDoS detection strategies.

The rest of the paper is organized as follows: Section 2 discusses related works. Section 3 reviews the state-of-the-art of DDoS detection techniques, focusing on taxonomies used to characterize these techniques. Section 4 presents the experimental framework developed to carry out the main performance evaluation

objectives of this study, including the benchmark datasets, the evaluation metrics, and the feature types. Section 5 describes four experiments and discusses the outcome of the comparative performance analysis. Section 6 summarizes the main contributions and findings of the proposed work, and discusses the future direction. Section 7 is the appendix section, which includes brief summaries of all reviewed DDoS detection techniques.

## 2. Related works

The literature contains a large number of early surveys, focusing on extensive classifications of both DDoS attacks and defense mechanisms. Mirkovic and Reiher proposed one of the earliest complete taxonomies of DDoS attacks and defense mechanisms [7]. The authors analyze the goal and strategies of DDoS attacks, and classify DDoS defense mechanisms based on activity levels, deployment locations, and degrees of cooperation. In [8], the authors analyze the causes of DoS attacks and present a detailed description of various DDoS defense schemes. They also compare the relative strengths and weaknesses of the reviewed schemes. The taxonomies provide deep understanding of the fundamental cause and shed light on traditional defense strategies against DDoS attacks. The use of intelligent techniques in DDoS detection, however, was not a main focus of these taxonomies. Recently, multiple surveys, focusing on different perspectives of DDoS attacks and defense mechanisms, were proposed [9–14]. In [9], the authors present an in-depth review of well-known DDoS defense schemes, focusing on the impact of deployment location on the performance of these schemes. The works discuss several issues critical to the effectiveness of DDoS defense strategies, and analyze the strengths and limitations of different schemes in addressing these issues. In [10], the authors provide an extensive classification of DDoS flooding attacks. A number of defense mechanisms, based on their deployment location, are reviewed. The strengths and limitations of these defense mechanisms are broadly discussed. Although informative, the use of intelligent mechanisms in these schemes is limited. In [11], the authors adopt the taxonomy proposed in [7] and use it to review a collection of DDoS detection methods. Future directions to detect DDoS attacks are proposed. In their work, machine learning based DDoS detection schemes are discussed. However, the authors only briefly describe those schemes. No performance or comparative analysis is provided. The work in [12] provides a taxonomy of DDoS attacks against cloud computing infrastructure. Several techniques to detect these attacks, based on deployment locations, are reviewed. In [13], DDoS defense is organized into three stages, namely prevention, detection and mitigation. Based on this organization, the authors highlight challenges in each stage and review the proposed solutions. In [14], the authors review several DDoS defense schemes that leverage the Software-Defined Network (SDN) architecture. They also discuss the vulnerabilities of SDN itself to DDoS attacks. The use of intelligent techniques to achieve efficient traffic analysis and detect attacks against the DNS is suggested. A discussion of these techniques to SDN-enabled networks is, however, lacking.

A number of surveys proposed in the literature have focused on network intrusion and anomaly detection [16–20]. Although these works did not sufficiently highlight features particular to DDoS attacks, they provide insights for possible solutions. In [19], the authors review anomaly detection methods from a statistical-, knowledge-, and ML-based perspective. They also discuss the challenges, merits and limitations of different assessment approaches and provide a list of publicly available traffic datasets. In [17], the authors review intrusion detection systems that applied computational intelligence methods. For each reviewed method, they provide a short tutorial and describe the corresponding application of the method to intrusion detection. Major ML methods, such as Decision Tree and Clustering, are not included in their work. In [18], the authors summarize a significant number of works that focused on Network Intrusion Detection Systems. They categorize these schemes according to their underlying technique, and provide a summary table of related references for each category. In [20], the focus is on data mining and ML schemes for network intrusion detection. The authors provide the technical details of a large number of machine learning methods. They also provide recommendations on how these schemes can be used to detect network intrusion, based on the characteristics of their attack and environment.

All aforementioned works mostly focus on the performance of specific defense methods. As such, they lack a comprehensive comparative analysis of the capabilities of ML-based DDoS detection schemes. Given the vast number and diversity of the proposed schemes, such an endeavor may not realistic. The intricate implementation details of these schemes further compound a comparative analysis of these schemes in a fair manner. Rather than focusing on the specifics of a scheme, a different approach is taken in this work to analyze the capability of the underlying building blocks that are commonly used in ML-based detection schemes. Based on this approach, an empirical evaluation of widely used techniques in intelligent DDoS detection is carried out.

## 3. Intelligent DDos detection taxonomy

Numerous schemes have been proposed to defend against DDoS attacks. Traditional DDoS defense methods, such as D-WARD [21], monitor and analyze the incoming and outgoing network traffic to detect malicious behavior. The network is considered under attack if the measured characteristics of the network traffic deviate significantly from those of what is considered to be normal traffic. The challenge stems from accurately differentiating between normal and anomalous behavior. Recent advances in ML hold promise for enabling effective capabilities to automatically learn the underlying data attributes, handle high-dimensional data, and support tolerance to errors. In this work, we focus on evaluating ML-based techniques for DDoS detection. To this end, we surveyed the literature and developed a taxonomy of intelligent DDoS detection techniques, which is shown in Fig. 1, along with the references. To select a representative set of ML-based techniques, a thorough search of the academic publications using Google Scholar, IEEE Xplore and Science Direct, is carried out. The search space was limited to the publishing period ranging from 2007 to 2018. Additionally, only publications with high citation numbers were considered. Using this process, 49 academic publications are included in this work. These publications are used to extract the building blocks underlying their DDoS detection schemes. Tables 8 and 9, provided in the appendices, summarize these publications. Furthermore, to gain

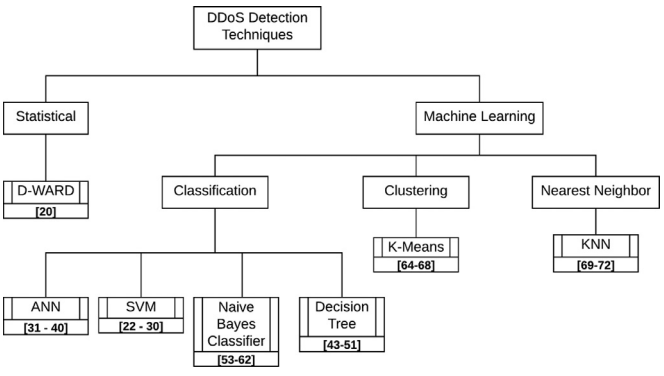


**Fig. 1.** A Taxonomy for DDoS detection techniques.

better understanding of the capabilities of ML-based techniques in comparison with traditional techniques, D-WARD [21], a statistics based DDoS defense scheme, is included in this study. In the following, taxonomy of intelligent DDoS detection mechanisms is briefly described and references relevant to each category are reviewed.

### 3.1. Classification for DDos detection

Classification is a supervised learning approach. The objective of a classification algorithm is to infer a mapping function, $f$, which maps input variables, $\vec{x}$, to the output class label, $y$, based on the labeled training data. The mapping function is then used to predict the classes a new observation belongs to. Intuitively, the DDoS detection problem can be modeled as a binary classification problem, where the monitored traffic is classified as either legitimate or attack traffic. The generic workflow, used by most intelligent DDoS classification schemes, is depicted in Fig. 2.

The workflow mainly consists of three sequential stages namely the *Data Preparation*, the *Training Phase* and the *Testing Phase*. The main objective of the data preparation stage is to transform the collected raw data into a well-formatted dataset of desired features and corresponding labels. For network traffic classification, the data is obtained from a number of sources, including captured network traffic, monitored network statistics, and sampled network packets. The collected data is then pre-processed. The preprocessing procedure may defer depending on the characteristics of the selected machine learning algorithm and the knowledge domain of the problem. Upon data processing, *Feature Selection*, a key component of a classification model, is carried out. The goal of feature selection is not only to accelerate the training process by reducing the number of attributes in the dataset, but also to improve the performance of the classification model by eliminating irrelevant attributes.

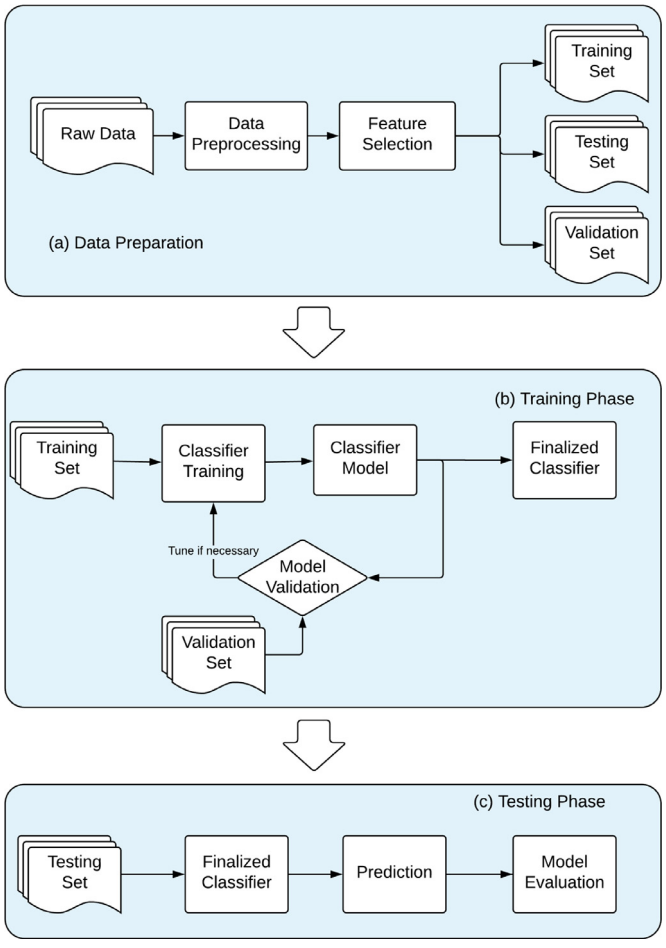


**Fig. 2.** A generic workflow of machine learning classification techniques.

The resulting dataset, represented by the desired set of features, is split into disjoint sets, which are used for training and testing phases. During the training phase, the selected learning algorithm uses the training set to build a classifier by automatically learning the model parameters. It is to be noted that most learning algorithms require that a set of hyper-parameters be configured manually. Determining the optimal values of a model's hyper-parameters for a given problem, however, is challenging. Different strategies have been used to select proper hyper-parameters, including rule-of-thumb, previous experiences, values used in other successful applications, and validation procedures. The tuning of the hyper-parameters using a validation set is illustrated in Fig. 2, which depicts a generic workflow of ML classification model. Aimed with different sets of hyper-parameters, the selected learning algorithm can be trained with the training set to produce different classifiers. A validation set, which does not overlap with the training set, is then used to estimate the performance of the produced classifiers. The specific set of hyper-parameters that achieves the best performance is used to build the finalized classifier. In the testing phase, the performance of the finalized classifier is evaluated based on the predictions it produces using the testing set.

The workflow described above is used to evaluate classification techniques for DDoS detection. Four classification techniques, namely Support Vector Machine (SVM), Decision Tree (DT), Artificial Neural Network (ANN) and Naive Bayes Classifier(NB), are evaluated.

**Support Vector Machine (SVM):** SVM attempts to solve an optimization problem that consists of finding a decision boundary in the feature space that separates data from different classes, while maximizing the distance from any data points to the boundary [22]. The distance from the decision boundary to the closest point is called the *Margin*. Data points that align the margin compose the *Support Vector*. For non-linearly separable features, the "kernel tricks" can be applied to transform input data into high dimensional feature vectors. SVM is then used to solve the optimization problem and identify the separating hyperplane.

The standard linear SVM and SVM with Radial Basis Function (RBF) kernel is frequently used to detect DDoS attacks [23–29]. The main difference among these methods is the selected features that characterize network traffic. In [23], the authors define an IP Address Interaction Feature (IAI), taking into consideration the proportion of packets sent to and received from the victim, the range of random ports used by the attacker, etc. The IAI is then used to build a linear SVM classifier. In [24], the authors observe that when a DDoS attack occurs, a large number of source IP addresses are sending packets to a few destination IP addresses. Additionally, contrary to flash crowd events, the source IP addresses may not have been seen by the victim before. Based on these observations, a set of features, defined by conditional entropy, is used to quantify the observed asymmetric property, and build the linear SVM classifier. In [27,29], the authors take advantage of Software-Defined Network (SDN) to defend against DDoS attacks. The proposed methods, the SDN centralized controller collects traffic information of the whole network and then builds a desired feature set. In [27], the authors use the entropy of source IP addresses and number of flows to build a SVM classifier. Similarly, in [29], the authors propose a linear SVM classifier, using the entropy information of the source IP address, destination IP address, and source and destination port numbers. In [25,26], SVM with the RBF kernel is applied to detect DDoS attacks. Features used in both schemes are extracted from network packets.

Additionally, attempts were made to extend the standard binary SVM to a multi-class classifier, aiming to distinguish different levels of attack intensities and different types of DDoS attacks [30,31]. In [30], the authors divide the intensity of attack traffic into four categories, namely normal, light, medium and heavy. A binary SVM classifier is built for each category pair. When unseen network traffic arrives, all classifiers are run separately, and the decision is then made using a majority voting rule. In [31], the authors propose a two-tier hierarchical detection scheme using multiple one-class SVMs. As a variation of the standard binary SVM classifier, one-class SVM is trained by learning data samples only from one class, namely the normal traffic class. During the testing phase, the classifier decides whether the testing instance is unlike normal traffic. In the proposed work, the first tier identifies attack traffic from legitimate traffic. The second tier, which is composed of three one-class SVMs, further classifies attack traffic into one of three types, TCP SYN flooding, UPD flooding and ICMP flooding.

Linear SVM and SVM with RBF Kernel (RBF-SVM) are widely applied in detecting DDoS attacks and have exhibited high-performance detection. Although not widely used in the reviewed literature, the Polynomial Kernel Function is inlduded in this study for completeness.

**Artificial Neural Network (ANN):** An ANN is inspired by the structure and functions of biological neural networks [32]. It is composed of a collection of nodes, called artificial neurons, which are organized into layers and interconnected by weighted edges. Fig. 3 represents a basic artificial neuron. For each neuron, the arriving signals are weighted summed and then transformed as an output signal by an activation function. Output signals are then passed to the next layer. This process continues until it reaches the last layer, referred to as the output layer. Layers between the input and output layers, where processing and computation are performed, are called hidden layers. The weights of connected neurons are initially randomly assigned and then optimized by the underlying learning algorithm, during the training procedure. Backpropagation is the most commonly used learning algorithm to optimize edge weights, using gradient decent. There are many types of ANNs. A basic ANN comprises a feedforward network connection, which does not contain cycles.

Different DDoS detection schemes have been proposed, mostly using a basic ANN with the Backpropagation algorithm [33–37]. The main difference among these schemes is the structure of the proposed ANN, in terms of the number of neurons in each layer and the number of hidden layers. Most reviewed ANNs have only one hidden layer. Neurons in the input layer usually represent the extracted features from network traffic, and the neurons in the output layer represent the desired labels. The number of neurons in the hidden layer typically ranges from 3 to 50.

ANNs are applied for different detection purposes. The work in [38] uses an ANN to estimate the number of zombies that are involved in a DDoS attack. To detect an attack, the system proactively constructs a normal profile, and continuously monitor the network traffic. When the entropy of flow size deviates beyond a predefined normal threshold, a DDoS attack is reported. The deviation value is input into the ANN model to estimate the number of zombies.

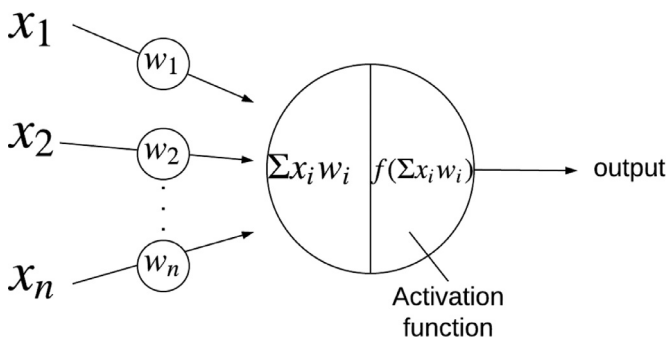


**Fig. 3.** A basic artificial neuron.

To improve the accuracy of detecting DDoS attack, an ensemble detection scheme, which combines multiple ANN classifiers, is used in [39]. In the proposed scheme, the training dataset was first partitioned into two classes, representing attack and normal traffic, respectively. The dataset in each class was further split into $n$ subsets. In each subset, the data was sub-divided into $k$ disjoint sets. Using these disjoint sets, $k$ training sets were re-constructed by leaving one of the disjoint sets out. As such, $k \times n$ ANN classifiers were built for each class. A new instance is then tested through all classifiers. The decision across $n$ subsets in each class is achieved using weighted majority voting, while the decision across different classes uses a weighted product rule.

Although Backpropagation is the most often used learning algorithm, other learning algorithms have also been applied in ANN models. In [40], the authors propose an ANN model for detecting HTTP flood attacks by using Genetic Algorithm (GA) to obtain the appropriate weights of the model. Initially, the GA randomly creates a population of solutions. Each solution is a set of weights that are required for the neural network. The fitness of each solution is measured by the output errors for training samples. Based on the fitness values, the general steps in GA, including selection, crossover and mutation, are processed and repeated until the highest fitness value produced the solution is greater than the desired score. The set of weights that have the highest fitness value are selected for the model.

In addition to using different learning algorithms, different types of ANNs are also used to detect DDoS attacks. In [41,42], the authors applied a RBF Neural Network for DDoS detection. A RBF Neural Network typically has three layers, namely an input layer, a hidden layer and an output layer. Neurons in the hidden layer are called RBF neurons. Intuitively, each RBF neuron stores a "prototype" of training samples, which represents the center of a local group that has similar training instances. When classifying a new instance, the distance between the testing instance and the prototype is measured and the this distance value is used as the input for the corresponding RBF neuron. Output signals from different RBF neurons, which constitute the RBF function of input values, are weighted summed as the output of the network. The training process consists of deciding the prototypes, selecting the parameter used in RBF function and optimizing the weights that connect RBF neurons and the output layer.

In this work, we implement a basic ANN, which has a feedforward architecture and uses the Backpropogation learning algorithm. The structure of the evaluated ANN has three layers, including a hidden layer, which are usually sufficient for a wide range ot application problems [43]. The hidden layers contain $n - o$ neurons, where $n$ is the number of features and $o$ is the number of output neurons. This decision is made based on our experience and the rule-of-thumb.

**Decision Tree (DT):** A DT is a tree-structured classifier, where each internal node represents a splitting rule and each leaf node represents an outcome. To construct a DT classifier, the algorithm recursively decides the most relevant feature and splits the training samples accordingly. To predict a new data sample, the algorithm visits branches in accordance to the features of the sample and makes a decision when a leaf node is reached. Different algorithms use different metrics to measure "the most relevant" feature. A comprehensive review of decision tree construction and application samples can be found in [44].

Contrary to other classifiers, such as SVM and ANN, the constructed DT model is easy to interpret, making it popular in various

applications. A set of rules can be directly obtained from the constructed tree model. Considering the DDoS detection, a DT model provides well-defined rules to accurately distinguish attack traffic from legitimate traffic.

DTs have been frequently used to detect different types of DDoS attacks [45–47]. In [45], the authors focus on flooding-based DDoS attacks. They develop a system that detects DDoS attack at the victim side and traces back the approximate geographical location of the attacker. Given sixteen features, which are all calculated from packet header information, a DT model is trained to classify traffic into one of four classes: normal, TCP SYN flooding, UDP flooding and ICMP flooding. When the attack traffic is detected, the traceback module is triggered. In [46], the authors focus on backscatter traffic. They employed DT and NB as traffic classifiers. Comparing the classification results, DT outperforms NB in both detection rate and false positive rate. Additionally, they define two sets of packet header features. Their results show that a high performance can be achieved without using IP addresses and port numbers as features. In [47], the authors apply DT to detect SYN flood attacks. The model is trained with five features, including destination IP address, TTL, frame size, MSS field and acknowledgement number, which are claimed to be related to SYN flood attacks.

In order to prevent DDoS attacks organized by botnets, a number of schemes utilize DT to detect the botnet communication traffic [48–50]. In [48], the authors apply DT to classify network traffic into different application fields (i.e. HTTPWeb application and BitTorrent) with a signature feature derived from packet payloads. In [49], the authors use DT to classify malicious and non-malicious traffic using a set of features selected from network flow-level, which captures the characteristics of a single flow for a given time interval. In [50], the authors propose a framework, which is built on Hadoop, to detect peer-to-peer botnet traffic. The proposed detection scheme uses a random forest, composed of 100 DT classifiers. Each DT classifier is trained with five features, randomly selected from a larger feature set, and only 1/10 of the whole training dataset. The framework carries out training and detection in parallel to improve performance.

Efforts to reduce the DT training and detection time, without sacrificing performance, have been proposed [51–53]. To achieve this objective, different feature selection methods are applied to reduce the feature size. In [51], the authors apply Information Gain and Chi-Square to rank features. The results show that the top ranked features identified by these two methods are the same. The impact of attribute size is also evaluated. The results show that the performance of the DT-based DDoS detection remains the same when the number of attributes is larger than nine. In [52], the authors use correlation coefficient feature selection for each network protocol. The selected features are used to train a DT classifier for each protocol. The detection results for different feature sizes are compared. The results show that that a negligible performance degradation occurs, when training the classifier with top ranked features. In [53], the authors apply four different features filter methods, namely Information Gain, Chi-Squared, Gain ratio and ReliefF, to rank features. The common top 1/3 features are then selected for training the DT classifier. The results show that the training time is reduced when the proposed feature selection procedure is used, in comparison with using the whole feature set.

Based on the reviewed literature, most schemes use the Information Gain to select relevant features and build the DT [45,47,48,51–53]. One work uses Gini Index to measure relevant features [46]. The rest did not provide detailed information of the metric used to construct the tree. In our work, we evaluated DT with both Gini Index (DT_Gini) and Information Gain (DT_IG).

**Naive Bayes Classifier (NB):** NB is a family of probabilistic algorithms, based on applying Bayes' theorem with strong independence assumptions between the features [54]. The model calculates the conditional probability of a given sample belonging to a specific class. To predict the label of a new observation, the class that has the highest probability with given input features is selected. Although the core assumption of input features conditional independence rarely holds true in the real world, NB performs surprisingly well for practical problems.

Formally, a NB classifier is to estimate $P(y|\boldsymbol{x})$, where $y$ is the a class variable and $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$ is a feature vector that represents a sample data. According to the Bayes theorem and the Naive Bayes assumption:

$$P(y|\boldsymbol{x}) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

Then, the class of a given instance $\boldsymbol{x_i}$ is assigned by:

$$\hat{y} = \text{argmax}_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

Training a NB classifier consists of estimating the prior, $P(y)$, and the distribution of attributes given class labels, $P(x_i|y)$, from the training data. Maximum Likelihood Estimation (MLE) and Maximum A Posterior (MAP) are two methods commonly used for estimation. Continuous attributes can be discretized into bins and transformed into discrete attributes [55,56]. Specific distribution, such as Gaussian distribution, can also be used for this purpose, whereby the estimation process consists of estimating the parameters of the distribution.

Based on the survey literature, it is widely believed that observable patterns that characterize DDoS attack events repeatedly occur over either a time window [57–59] or a packet-counting window [60]. In these works, the authors split the input traffic into subsets, called windows, based on a time period or the number of arriving packets. During the testing phase, the detection is completed by consulting the probability table, whose entries are estimated based on previous windows.

NB classifiers have also been used to detect different types of DDoS attacks. In [56,61], the focus is on detecting application layer DDoS attacks. Both works utilize the network flow-based features to train a NB classifier. In [62], a layered approach is proposed. Each layer is an individual NB classifier, and takes responsibility of one type of attacks. Features are selected specifically for each type of attacks. The approach presented in [63] uses the power spectral density and Haar wavelet coefficients as features to train a NB classifier for DDoS attack detection.

Efforts have been made to improve the performance and efficiency of NB classifiers for DDoS detection [55,64]. In [55], the authors propose an algorithm to adaptively reduce attributes in order to design a NB classifier with zero training error. During the training procedure, after the probabilities required by the NB classifier are estimated, the same training data is classified with the estimated probabilities. If an error is detected, the information gain is applied to all features, and the most relevant feature is selected to divide the dataset into two parts. The process of training NB is repeated in the two sub-datasets, with one feature eliminated. This training and dataset splitting procedure is repeated until no error occurs. In [64], researchers propose methods to improve the efficiency of training the NB classifier. Since the estimation can be done independently, the authors parallelize the training procedure and show that the training time is reduced significantly.

In our evaluation, we used MAP to estimate the $P(y)$ and $P(x_i|y)$. We assumed the Gaussian distribution for continues features. Most of reviewed articles provide a vague description about the detailed techniques applied for estimating probabilities and the detailed procedure applied to process features.

### 3.2. Clustering for DDos detection

Clustering is an unsupervised learning approach, which aims to find natural groups of observations based on the inherent similarities within a dataset. Theoretically, data points that are in the same group should share similar properties, while data points that are from different groups should have high dis-similarities. A generic workflow for clustering is depicted in Fig. 4.

The first step in clustering is to pre-process and transform the raw data into a well-formatted data set for analysis. Distance measures, also referred to as similarity measures, must be appropriately defined. Using prior knowledge of the application domain or a specific clustering objective, distance measures can be derived from a variety of measurements that have been applied in various research fields, a few of which are listed in Fig. 4. A new distance function tailored to a specific problem domain can also be derived. Given the formatted dataset and a distance measurement, clustering algorithms are then applied. Since labels are not associated with samples, the output clusters usually must be further analyzed to understand the represented groups. In the case of DDoS detection, upon successfully separating normal and attack traffic into different clusters, post-analysis is still required to indicate which cluster represents the legitimate traffic and which cluster represents the attack traffic. Additionally, most clustering algorithms require all data samples to be available. These two limitations requires that the analysis be done off-line, which makes most clustering algorithms unsuitable for online DDoS detection. Attempts have been made o overcome these limitations and use clustering as building block for DDoS attack detection. In this work, we evaluate one clustering-based technique, K-Means, widely used for DDoS detection.

**K-Means**: K-Means algorithm, also referred to as Lloyd's algorithm, is one of the best-known clustering algorithms [65]. Initially, the algorithm randomly picks $k$ centroids, one for each cluster. Then the algorithm assigns each data to one cluster based on its distance to the $k$ centroids. After all observations have been assigned to one cluster, the centroids are re-calculated. The procedure of assigning data and calculating centroids is repeated until the assignment no longer changes.

To address the limitation of the K-Means, as a clustering algorithm for DDoS detection, the process of assigning labels to the resulting clusters has been proposed [66–68]. In [66], the authors apply the post-analysis to decide labels for clusters. They investigate a particular range of feature values in each cluster, then label the cluster as normal, pre-attack or attack. In [67], the author proposed an offline detection scheme for botnet traffic. They provide three methods to label the resulted clusters. In the first method, $k$ was set to two, so all samples are grouped into two clusters. The cluster that has the smaller standard deviation is labeled as botnet traffic. In the second method, $k$ is a variable that can be adjusted by users. The label of each cluster is determined by both the standard deviation and the proportion of the number of normal/attack instances to the total number of normal/attack instances. The third method also allows to generate multiple clusters, then the label of each cluster is determined by the label of the majority members. In [68], the author proposes a detection scheme for application layer DDoS attacks. Before applying K-Means clustering, the system filters out users that are believed to be normal. Then the rest of records, referred to as suspicious users, are used as input to K-Means clustering. The K-Means algorithm separates them into two clusters, and the cluster, which has more members, is labeled as the attacker group.

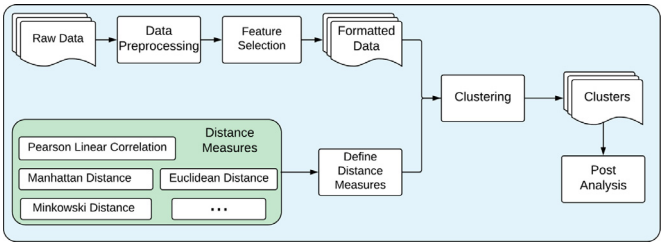


**Fig. 4.** A generic workflow for machine learning clustering techniques.

In [69,70], a K-Means-based clustering algorithm as a classifier for online detection of DDoS attacks. During the training phase, the authors apply K-Means clustering on normal traffic only. The centroids of normal clusters is then used for DDoS detection. Upon the occurrence of a new traffic instance, the model computes the distance from the new sample to each memorized centroid. If the distance is beyond a certain range, the instance is classified as an attack traffic. Different approaches are used to compute the range. In [69], the radius and the proportion of the number of members in the cluster to the total number of members are recorded, for each normal cluster's centroid. If the distance between the new instance and the closest centroid is within the radius, then the sample is labeled normal. Otherwise, a trust score is calculated based on the distance, the radius and the proportion. The trust score is then used to label the sample. In [70], the scheme calculates the distance between the new sample and all normal clusters' centroids to class iffy a new instance. If the distance to all normal clusters is beyond the allowed threshold, then the sample is labeled as abnormal.

In our evaluation, we adopt the second strategy, which applies K-Means as a classifier. During the training phase, the K-Means algorithm is applied to legitimate traffic and attack traffic separately, to ensure no false clustered instances. For each type of traffic, K-Means algorithm generates eight clusters, for a total of sixteen clusters. The centroid of each cluster is stored. Upon the occurrence of new instance, the distance between the instance and each centroid is calculated. The new instance is then assigned the label of the closet cluster. The decision of the value $k$ was not discussed in all reviewed articles. We choose $k = 8$ to generate a "tighter" cluster, which is also the suggested default setting by the python scikit-learn package.

### 3.3. Nearest-Neighbor for DDoS detection

Nearest-Neighbor based techniques classify samples based on the similarity of the population individuals. Contrary to previously discussed classification techniques, Nearest-Neighbor based techniques do not learn a mapping function between features and labels. The technique simply memorizes all training instances. When an unseen instance arrives, the model measures the distance between the new instance and all stored samples. the distance is then used to label new instances. A generic workflow of Nearest-Neighbor technique is depicted in Fig. 5.

The first step, *Data Preparation*, data are processed in a similar manner as in classification work flows to produce well-formatted training data. A distance function is then used to measures the similarity between samples. Not, however, that no training phase is required, since all training samples are memorized by the designed scheme. During the testing phase, the unseen sample is compared with the memorized training samples, and the label will be assigned based upon the similarities. In this work, we evaluate the K-Nearest Neighbors technique.

**K-Nearest Neighbors (KNN)**: KNN is widely renowned for its effectiveness. It classifies an unseen data by calculating the distance between the data and all training samples. It then finds the $k$ most closest training samples, and labels the unseen data using a majority vote rule.

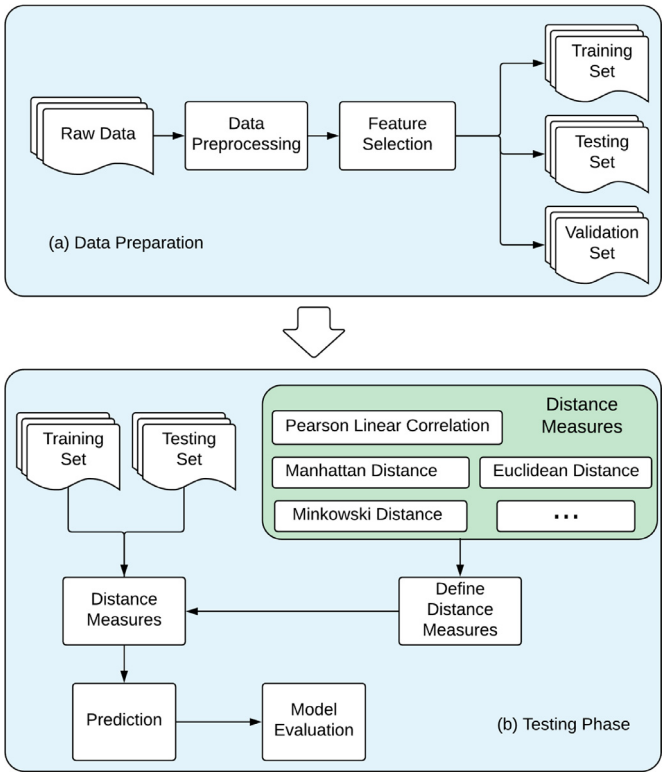


**Fig. 5.** A generic workflow for nearest-neighbor techniques.

Several KNN-based DDoS detection schemes have proposed. These schemes mostly differ in the way they explore different feature spaces to construct the detection model [71–74]. In [71], the authors compute the entropy of the source and destination IP addresses, source and destination Port numbers, number of packets, etc. These entropy values are then used to measure the similarities among data samples for the KNN classifier. The trained KNN classifier is then used to detect the different network states, including a normal state and when the network is under DDoS pre-attack and attack. In [72], the authors chose thirty-five features derived from packet headers captured within a two-second time window. KNN is then used as a classifier to detect the likelihood of an attack during the two-second time window. In [73], the authors observe that network flows generated by the same application are correlated. Based on this observation, they apply KNN with network flow level features to classify the network traffic into one of five applications, namely HTTP, Bit Torrent, DNS, Hadoop and DoS. To reduce the high computational overhead of running KNN, the authors also propose a grid-based spatial index to reduce the number of training samples involved in the classification stage. In [74], the authors also use features extracted from network flow information. They claimed that the information, such as median of packets per flow, median of bytes per flow, etc., can capture the network state changes, during an attack. To preserve privacy and improve the efficiency of KNN, a combined data and perturbation encryption approach is used to reduce computation time, without loosing accuracy.

As described previously, to measure the similarity, a distance function is applied. Various distance functions have been used in the literature, although the Euclidean distance by far the most popular one. Hence, Euclidean distance is used in our implementation. Additionally, the value of $k$ must be determined. Determining an optimal $k$ that can be used for a comparative analysis of different algorithms is challenging, as the value varies depending on the problem domain and application. Mindful that the objective of the paper is not to optimize the performance of a specific algorithm, but rather to carry out a fair comparison, the default value, $k = 5$, is used in out implementation.

### 3.4. Statistical – D-WARD

D-WARD is a statistical detection method to defend against DDoS Attacks [21]. Statistical methods usually assume a predefined distribution to model normal behavior. The basic tenet of D-WARD is continuous monitoring of the network traffic to infer the behavior of the network traffic. The inferred behavior is then compared to the predefined normal behavior to determine non-complying traffic. Non-complying traffic is considered attack traffic. D-WARD is designed to be deployed in a border router at the source-end of the attack. It is composed of a detection and throttling components. The detection component continuously monitors the two-way traffic between the policed network and the rest of the Internet. Periodically, the detection scheme derives the statistics of the monitored traffic and compares them to those of the predefined normal traffic. Upon detection of anomaly traffic, the throttling component is triggered to mitigate the impact of attack traffic. The focus of this study is the detection component. As such, the efficiency of the throttling component is not considered. The suggested settings for all parameters required by the detection component are used in our study.

## 4. Experiment framework

### 4.1. Datasets

In order to carry out a meaningful and fair comparative analysis, a benchmark that closely reflects the real-life network traffic, with clearly labeled legitimate and attack traffic, is required. Generating such a benchmark presents a significant challenge. One approach is to use simulation to generate such a benchmark, and clearly label legitimate and attack traffic. The drawback of this approach is that it may fail to accurately model real-world scenarios, which may potentially lead to a biased comparative analysis. A second approach consists of capturing real network traffic. The drawback of the latter approach is that it may not always be possible to label correctly all traffic, especially traffic generated by unknown attacks. Consequently, techniques, which successfully label unknown attack traffic, may unfairly be penalized. To address these shortcomings, the proposed approach is to combine two widely accepted and extensively used benchmarks.

The first benchmark is CAIDA DDoS dataset, from an actual DDoS attack event [75]. The most significant advantage of this dataset is that the traffic consists of real-world DDoS attack scenarios. Moreover, DDoS attack traffic is exclusively recorded in this dataset, thereby enabling effective evaluation of DDoS attacks. To construct the desired benchmark, which both captures the network characteristics and enables fair evaluation of DDoS detection schemes, the CAIDA dataset is augmented with the DARPA dataset of legitimate traffic [76], is used. The DARPA dataset contains traffic collected over a period of five weeks. The traffic collected during the first and third week contains no attack traffic.

The attack traffic, obtained from the CAIDA benchmark, is split into 14 sets, each containing roughly 5 min worth of traffic traces. The legitimate traffic is selected from DARPA's first week attack-free dataset. We use Tcpreplay [77] to replay the attack and legitimate traffic, and recapture the combined traffic for analysis. Let $D = \{d_k(I), 1 \leq k \leq 14\}$ denote the traffic datasets, where $d_k(I)$ represents the traffic captured during the $k$th time interval of size of $I = 5$ min.

Fig. 6 displays the ratio of attack traffic to legitimate traffic, in terms of average packet rate (packets/second) and byte rate

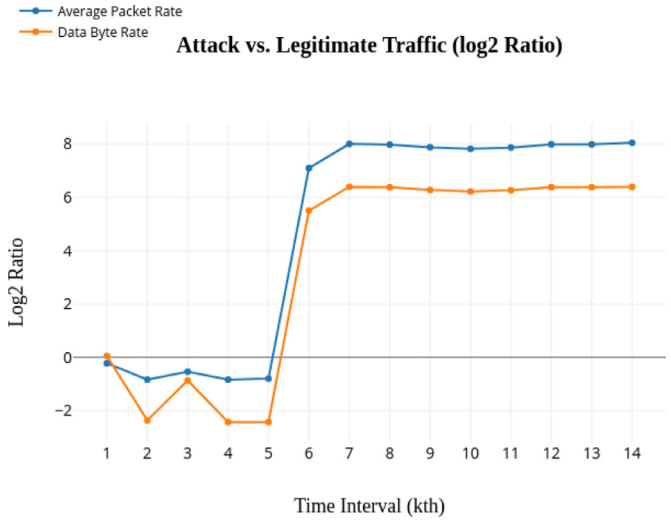


**Fig. 6.** Log2 Ratio of Attack traffic vs. Legitimate Traffic. The figure represents the ratio of attack traffic and legitimate traffic, in terms of average packets rate (packets/second), shown in blue, and data Byte rate (Bytes/second), shown in yellow. The log 2 scale is applied to display the large range without compressing small values..(For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

**Table 1**
Confusion matrix.

| | | Prediction | |
|---|---|---|---|
| | | Attack | Legitimate |
| Known | Attack | TPositive | False Negative |
| | Legitimate | FP | TN |

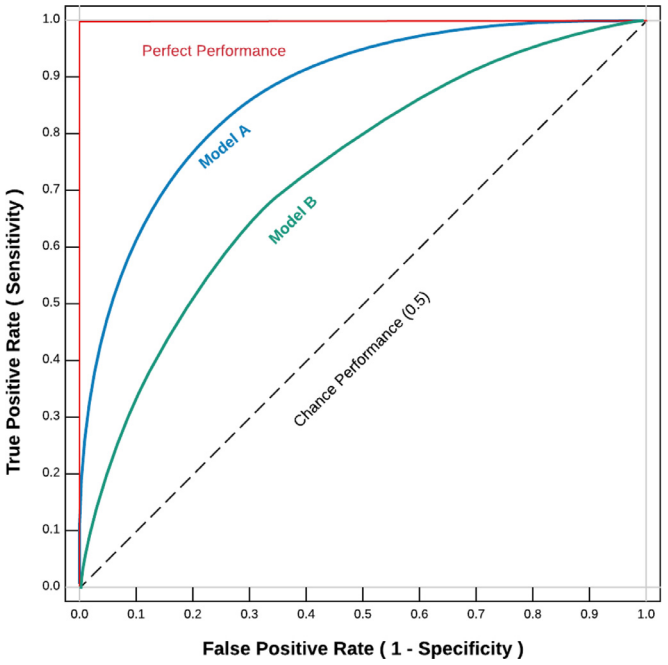


**Fig. 7.** A sample of ROC curve. The figure represents a sample ROC curve. It presents the trade-off between sensitivity and specificity. A perfect test, shown in the red curve, has an AUC of 1.0. The closer the curve to the left-top corner, the better the test. In this graph, Model A outperforms Model B. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

(Bytes/second). For $k < 6$ (), the average legitimate packet and byte rates, during the first 25 min of the data capture process, are higher than those of the attack traffic, which translates into negative values in a $log_2$ scale. Starting from $k = 6$ (the sixth 5 min period), the attack traffic volume increases dramatically. The changing traffic patterns depict different DDoS attack phases. In all experiments, the focus is on TCP and ICMP protocols, where a significant amount of attack traffic was observed. UDP traffic is discarded due to the limited number of UDP attack traffic observed in the CAIDA dataset.

### 4.2. Evaluation metrics

The DDoS detection problem is usually mapped to a binary classification problem, where the labels are by convention, positive (attack) and negative (legitimate). As such, the performance of a detector can be summarized by a confusion matrix using four outcomes namely, True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). The matrix is depicted in Table 1.

Based on the above matrix, the following five metrics are widely used to evaluate the performance:

- ***Accuracy***: $\frac{TN+TP}{TN+TP+FN+FP}$
- ***Sensitivity***, also known as ***Recall*** or ***True Positive Rate*** (**TPR**): $\frac{TP}{TP+FN}$
- ***Specificity***, also known as ***True Negative Rate*** (**TNR**): $\frac{TN}{TN+FP}$
- ***Precision***, also known as ***Positive Predictive Value*** (**PPV**): $\frac{TP}{TP+FP}$
- ***Fall-out***, also known as ***False Positive Rate*** (**FPR**): $\frac{FP}{TN+FP}$

These metrics are usually used complementary to assess the performance of a detector. For example, one popular and common evaluation approach in ML-based techniques is using Precision and Recall for evaluation. Due to the inverse relationship between these two metrics, it is rare to discuss Precision and Recall in isolation. Neither metric alone can be used to capture the performance of handling the legitimate traffic (negative samples) in DDoS detection. Furthermore, these metrics can easily introduce biases, which must be clearly understood for a fair analysis [78].

Another popular evaluation method is the Receiver Operating Characteristics (ROC) analysis, which is based on TPR and FPR. Since FPR is equal to (1 - Specificity), the ROC space illustrates the trade-off between Sensitivity and Specificity. A pair of TPR and FPR score determines a single point on the ROC space. In order to plot the ROC curve, the threshold that is used to determine the label of examining samples is varied. Hence, the ROC curve is created by plotting the TPR vs. FPR over its entire operating range, along with varied thresholds. Additionally, the ROC curve provides the ability to intuitively compare multiple models, as illustrated in Fig. 7. To reduce the two-dimensional depiction in the evaluation, Area Under the Curve (AUC) i is typically used [79]. However, for clustering methods or nearest neighbor based methods, which applied binary decision without using a threshold or generating a score, ROC analysis cannot be applied. The testing result will be a single point on the ROC space, as explained previously.

In this study, we use Accuracy, Sensitivity and Specificity to assess the performance of DDoS detection schemes. Accuracy measures the ratio between the correctly identified traffic and the total traffic. Sensitivity and specificity measure how well the technique performs for one category (legitimate or attack), whereas accuracy measures how well the technique performs for both categories.

In addition to evaluating the detection performance, we also study the impact of influential factors. To quantitatively evaluate the influence level, the Pearson Correlation Coefficient test is applied. The correlation coefficient score is interpreted based on the rule of thumb, shown in Table 2.

**Table 2**
Correlation coefficient interpretation.

| Correlation coefficient | Strength |
|---|---|
| −0.1 to 0.1 | None or very weak |
| −0.3 to −0.1 or 0.1 to 0.3 | Weak |
| −0.5 to −0.3 or 0.3 to 0.5 | Moderate |
| −1.0 to −0.5 or 1.0 to 0.5 | Strong |

### 4.3. Features sets

DDoS detection schemes usually collect network traffic through passive network monitoring. The collected traffic is then analyzed to identify attack traffic. There are two general approaches for passive network monitoring. One approach is packet capture, which intercepts and logs network data packets. Various tools can be used to gather data packets, including tcpdump [80], and wireshark [81]. The other approach is network flow monitoring, which provides aggregated traffic statistics for a flow between two end points [82]. As such, two feature sets, packet- and flow-level based, are considered for analyzing the performance of DDoS detection techniques.

Table 3 summarizes the packet- and flow-level based features. In this study, a flow is identified as a unidirectional sequence of packets, which share the same 5-tuple values, namely source IP address, source Port number, destination IP address, destination Port number, and Protocol ID.

In this work, the detection performance of ML-based techniques on both feature sets are studied. Since D-WARD requires specific features for the detection, we do not apply different feature sets to D-WARD.

## 5. Experiments and results

In this section, we present four experiments used to assess the performance of the different ML-based techniques to detect DDoS attacks. The first experiment is a comparative analysis, aiming to answer the question of which technique, if any, outperforms all others. To enhance our understanding of the impact of different influential factors, we conducted three additional experiments to analyze the performance of the DDoS detection techniques affected by the observable network traffic proportions, the attack traffic intensities and the imbalance dataset problem. In the following subsections, each experiment is described, and the results of these experiments are then discussed.

### 5.1. Exp. 1 – Comparative analysis

The goal of experiment 1 is to conduct a comparative analysis of the overall performance of the selected techniques. In this experiment, we divide the generated benchmark datasets, $D$, into $T_r$ and $T_s$, for training and testing purposes, respectively.

Classification models are usually trained offline with historical data. The trained model is then applied online to classify future data. In our benchmark datasets, $d_1(I)$ contains the network traffic that is captured during the first 5 min of the monitoring interval. It is considered to be the historical data and used to form $T_r$. $T_s$ is composed of traffic captured over the remaining 13 5 min intervals. Specifically, $T_r = \{d_1(I)\}$; $T_s = \{d_2(I), d_3(I), d_4(I), \ldots, d_{14}(I)\}$. During the testing phase, the 13 traffic sets are tested independently for each evaluated technique, using the accuracy, sensitivity and specificity metrics. Boxplots are used to illustrate the assessed performance, so that the performance variation of these evaluated DDoS detection techniques is also quantified.

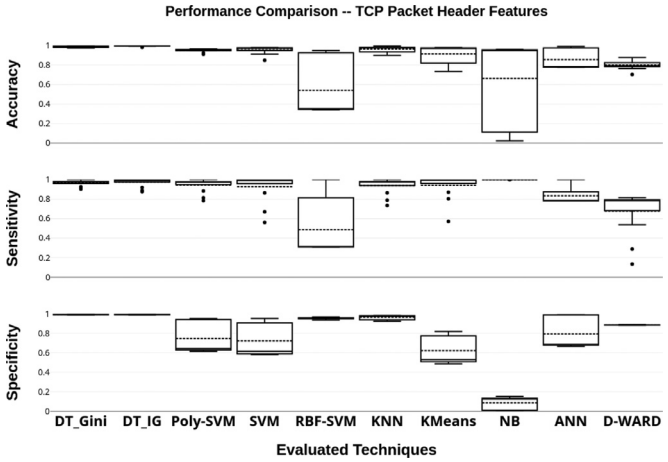


**Fig. 8.** Performance comparison – TCP traffic using packet header features.

Additionally, in this experiment, both packet-header and flow-level based feature sets are studied, and the impact of different feature sets are discussed. Since TCP and ICMP protocols have different packet-header features, and D-WARD deals with different protocols separately, the following subsections present and discuss the detection results per protocol. For the detection scheme of D-WARD, we use suggested default values for all parameters.

#### 5.1.1. TCP traffic

Results for TCP traffic using packet-header features are shown in Fig. 8. The results show that DT outperforms all other techniques, with respect to the defined performance metrics. The performances of the two different attributes split criteria for building the DT model, Information Gain and Gini Index, are comparable. Furthermore, the results show that DT performs consistently across different testing sets. These results show the high potential of rule-based strategies to efficiently detect DDoS TCP attack traffic in the packet level. In this experiment, NB and RBF-SVM exhibit the worst performance when it comes to distinguish attack traffic packets from legitimate traffic packets.

The NB classifier is a simple classifier based on Bayes' theorem, assuming a strong independence among the features. As such, given the packet-header features, the NB classifier is biased towards labeling most samples as attack traffic during the testing

**Table 3**
Feature sets description: Detailed contents of packet-header based and flow-level based features for TCP and ICMP.

| Feature Types | Protocols | |
|---|---|---|
| | TCP | ICMP |
| Packet Headers | srcIP, srcPort, dstIP, dstPort, Len, TTL, Seq, FIN, SYN, RST, PSH, ACK, URG, ECE, CWR, checksum | srcIP, dstIP, Len, TTL, Type, Code, icmpID, checksum |
| Network Flows | duration, rtt, protocol, srcIP, srcPort, dstIP, dstPort, iflags, uflags, isn, tag, pkt, oct, end-reason | |

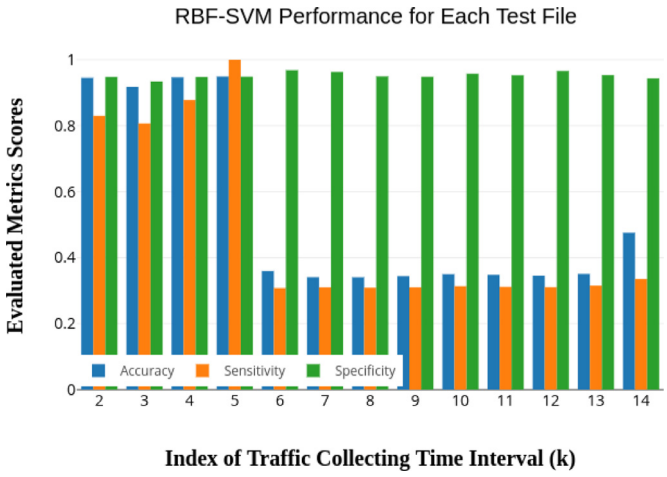


**Fig. 9.** RBF-SVMperformance using packet header features. X-axis presents the index of the time interval ($k$), and Y-axis presents the evaluated metrics values.

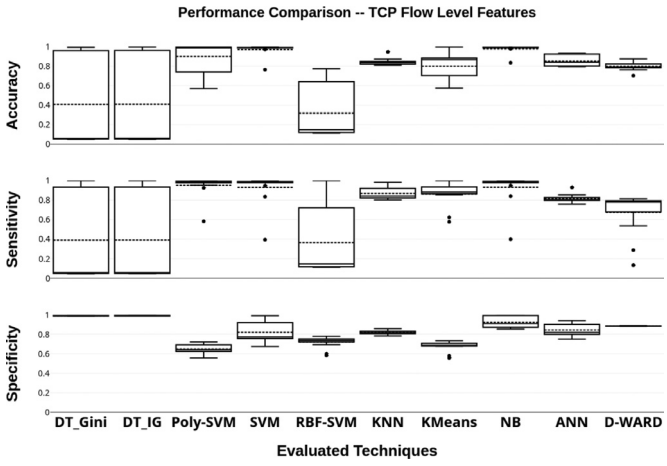


**Fig. 10.** Performance comparison – TCP traffic using flow level features.

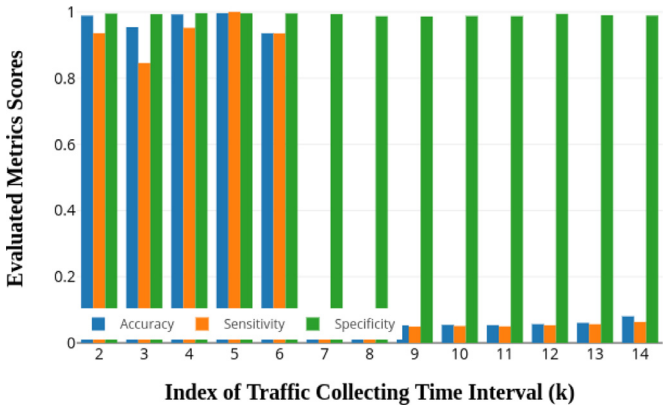


**Fig. 11.** DT-IG performance using flow level features.

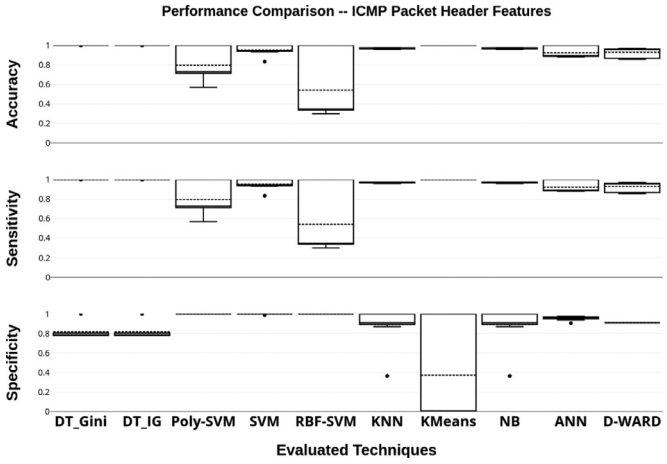


**Fig. 12.** Performance comparison – ICMP traffic using packet header features.

phase. This results in an extremely high sensitivity score, but a poor specificity score.

The RBF kernel maps features into an infinite dimensional space to solve non-linearly separable samples. This may lead to a loss of generalization, if the training samples are underrepresented. In other words, the trained model fits the training samples too closely, causing the model become very sensitive to the input data. Fig. 9 presents the performance of RBF-SVM in each test case. For the first four test cases $k$, $k = 2, 3, 4, 5$, RBF-SVM shows a good performance. However, for test cases $k$, $k \geq 6$, its accuracy and sensitivity are significantly decreased. Recall that in the attack event, presented in Fig. 6, the attack volume increases significantly starting from $k = 6$. The increase in volume affect the distribution of traffic attributes, which in turn causes the underlying patterns of the input dataset to become significantly different from that was learned during the training phase.

Focusing on the sensitivity score, the results achieved by SVM, Poly-SVM, KNN and KMeans, shown in Fig. 8, are comparable to those achieved by DT. They all outperform D-Ward. However, the poor specificity scores of these ML-based techniques suggest a potentially high rate of false alarms, which can incorrectly prevent legitimate users from accessing resources.

Results for TCP traffic using flow-level features are shown in Fig. 10. DT no longer demonstrates the observed superiority over all other ML-based techniques. Moreover, its accuracy and sensitivity scores are highly variable across all datasets. Examining closely to its performance on each test case, shown in Fig. 11, DT suffers the over-fitting problem, presenting a similar pattern of the RBF-SVM behavior. Additionally, using flow-level features, which represent features from the aggregated packet data between a source and a destination, significantly improves NB's accuracy and specificity. The performance consistency of NB across multiple test cases is also enhanced by using flow-level features. In comparison with packet-header features, flow-level features provide statistic attributes that capture the traffic behavior. Hence, the distribution assumption and the probability inferring of NB classifier is more reasonable.

Finally, it is worth noting that the traditional method, D-WARD, performs competitively in comparison to ML-based techniques. However, it failed to identify attack traffic in two datasets, as indicated by the poor sensitivity score for these two cases. Overall, it can be concluded that ML-based techniques can achieve high performance in detecting TCP attack traffic. Furthermore, ML-based techniques outperform D-WARD in most test cases.

### 5.1.2. *ICMP traffic*

Results for ICMP traffic using packet-header features are shown in Fig. 12. The results show that all ML-based techniques achieve near optimal values with respect to all performance metrics, but only for some datasets. Contrarily, the performance consistency of D-WARD is relatively higher. For ICMP traffic, D-WARD detects attack by monitoring the paired messages of ICMP requests and the corresponding relies. If the monitored ratio exceeds the

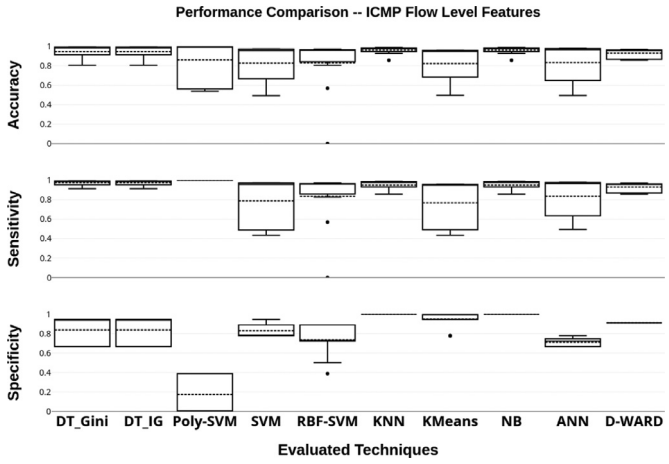


**Fig. 13.** Performance comparison – ICMP traffic using flow level features.

threshold, the alarm is raised. The results show that this simple mechanism works effectively. Features that capture this type of information should be able to improve the performance of ML-based techniques.

Results for ICMP traffic using flow-level features are shown in Fig. 13. The results exhibit a similar performance pattern as the one in TCP traffic, underscoring the fact that using sophisticated features does not necessarily improve the performance of all ML-based techniques. As indicated by the results, only specific ML-based techniques, such as Poly-SVM and NB, are improved. The use of these features did not significantly improve the performance of the other schemes. In some cases, the performance of these latter schemes has decreased.

In summary, the outcome of this experiment shows that it is not clear that a single technique outperforms all others in all test cases, especially when focusing on the ICMP traffic. The experiment shows that different techniques perform better when using certain types of features, suggesting that feature selection should be method specific. Furthermore, the capability of detecting attack traffic shown by ML-based techniques is evident. On the other hand, the performance inconsistency exhibited by ML-based techniques in dealing with different types of attack traffic raise doubts about their ability to efficiently detect DDoS attack in real world scenarios.

### 5.2. Exp. 2 – Impact of observable traffic proportions

In experiment 1, the entire network traffic is assumed to be available for each DDoS detection scheme. In practice, however, it is infeasible for a detection scheme to have access to the entire network traffic. Actually, the detection scheme is usually deployed on, or attached with, routers or switches. Consequently, a detection scheme can only observe the network traffic passing through the network device on which it is deployed. The purpose of Experiment 2 is to investigate the ability of a detection technique to only access a limited portion of network traffic.

To emulate a realistic network environment, we randomly select a proportion $p$ of the total traffic for testing. The selected traffic is then analyzed by the detection techniques, and the performance for each metric is evaluated. Two selection criteria, namely packet- and flow-level, are used to generate a specified portion, $p$, of the network traffic, For packet-level, a proportion, $p$, of the total traffic packets is randomly selected without any consideration of the flows to which the selected packets belong. For flow-level, however, a proportion, $p$, of the total traffic flows is randomly selected and only packets belonging to these flows are made available to the DDoS detector. It is to be noted that the total number of packets selected is different for each case.

**Table 4**
Correlation coefficient scores – % total traffic packet.

| Techniques | Correlation Score with | | |
|---|---|---|---|
| | Accuracy | Sensitivity | Specificity |
| DT-Gini | −0.0024 | −0.0123 | −0.0117 |
| DT-IG | −0.0103 | −0.0103 | 0.0 |
| SVM | −0.0002 | −0.0074 | 0.0006 |
| RBF-SVM | −0.0002 | −0.0022 | 0.0057 |
| Poly-SVM | −0.0003 | −0.0065 | 0.0008 |
| KNN | −0.0016 | −0.0088 | −0.0024 |
| KMeans | −0.0011 | −0.0058 | −0.0008 |
| NB | $< 0.0001^*$ | 0.0215 | −0.0011 |
| ANN | 0.0001 | −0.0012 | 0.0008 |
| D-Ward | 0.5458 | 0.8008 | −0.5519 |

* The value is negative, and $\|value\| < 0.0001$.

**Table 5**
Correlation coefficient scores – % total traffic flows.

| Techniques | Correlation Score with | | |
|---|---|---|---|
| | Accuracy | Sensitivity | Specificity |
| DT-Gini | −0.2698 | 0.0232 | −0.3496 |
| DT-IG | 0.0549 | −0.0159 | 0.0253 |
| SVM | −0.0177 | 0.2045 | 0.0147 |
| RBF-SVM | −0.2732 | −0.2384 | 0.2769 |
| Poly-SVM | −0.2473 | 0.0666 | −0.0661 |
| KNN | −0.1760 | 0.0885 | 0.01744 |
| KMeans | −0.0912 | 0.0300 | −0.1349 |
| NB | −0.0946 | 0.1870 | −0.4023 |
| ANN | −0.1283 | −0.0456 | −0.0857 |
| D-Ward | 0.7703 | 0.8055 | 0.6329 |

The same training data set $T_r$, used in experiment 1, is also used for this experiment. The traffic proportion $p$ is selected from a set $P = \{1\%, 5\%, 10\%, 20\%, 50\%, 75\%\}$. We do not use separate symbols to differentiate packet- and flow-level datasets, since they are structurally the same. The random traffic selection of DDoS detector observed traffic is repeated 10 times, with different random seeds, to avoid data bias. The dataset used for testing is denoted by $D_{exp2}$, where $D_{exp2} = d^p_{k,r}(I),\ 1 \le r \le 10,\ p \in P$ and $2 \le k \le 14$. In total, 780 test cases are used to assess the performance of each technique, for both packet- and flow-level selections.

To quantitatively evaluate the impact of the observable traffic proportions, Pearson Correlation Coefficient test is applied to measure the strength of the linear correlation between each evaluated metric and the observed proportion. For both packet- and flow-level traffic selections, the performance of D-WARD presents a strong correlation with the increasing observed traffic proportion. Conversely, only a weak correlation between traffic proportions and performance is exhibited by ML-based techniques. Tables 4 and 5 show results for packet- and flow-level traffic selections, respectively.

Recall that traditional detection techniques, represented by D-WARD, usually infer network status through monitoring two-way traffic. As such, D-WARD gains a relatively complete picture of the network status given a higher proportion of observed traffic packets or flows. This leads to more accurate attack detection. Ideally, D-WARD should be deployed at the only boarder router, so that both directions of flows can be observed by the detector. However, this is impractical, a limitation also reported by the authors of D-WARD [21]. Comparing to the traditional detection method, the weak correlation, presented by ML-based techniques, shows that the deployment location does not impact the performance of ML-based detection techniques.

**Table 6**
Correlation coefficient scores – attack intensity (in packets).

| Techniques | Correlation Score with | | |
|---|---|---|---|
| | Accuracy | Sensitivity | Specificity |
| DT-Gini | −0.2880 | −0.0098 | < 0.0001 |
| DT-IG | 0.2901 | −0.0103 | < 0.0001 |
| SVM | 0.5777 | −0.0039 | < 0.0001 |
| RBF-SVM | −0.4309 | −0.0023 | < 0.0001 |
| Poly-SVM | 0.5314 | −0.0026 | < 0.0001 |
| KNN | −0.0013 | −0.0079 | < 0.0001* |
| KMeans | 0.6209 | −0.0003 | < 0.0001* |
| NB | 0.3873 | 0.0117 | < 0.0001* |
| ANN | 0.0795 | −0.0010 | < 0.0001 |
| D-Ward | 0.5728 | 0.7936 | < 0.0001* |

* The value is negative, and $\|value\| < 0.0001$.

**Table 7**
Correlation coefficient scores – attack intensity (in flows).

| Techniques | Correlation Score with | | |
|---|---|---|---|
| | Accuracy | Sensitivity | Specificity |
| DT-Gini | −0.2862 | −0.0759 | < 0.0001 |
| DT-IG | 0.2856 | −0.0657 | < 0.0001 |
| SVM | 0.5777 | −0.0100 | < 0.0001 |
| RBF-SVM | −0.4309 | −0.0045 | < 0.0001 |
| Poly-SVM | 0.5314 | −0.0070 | < 0.0001 |
| KNN | −0.0008 | −0.0255 | < 0.0001* |
| KMeans | 0.6209 | −0.0033 | < 0.0001* |
| NB | 0.3875 | 0.0568 | < 0.0001* |
| ANN | 0.0795 | −0.0069 | < 0.0001 |
| D-Ward | 0.5636 | 0.8101 | < 0.0001* |

* The value is negative, and $\|value\| < 0.0001$.

### 5.3. Exp. 3 – Impact of attack intensities

In experiment 2, the focus was on the proportion of network traffic observed by a detector. In this experiment, we further refine the previous experiment to focus exclusively on the intensity of the attack traffic observed by the detector. Consequently, legitimate traffic is kept unchanged, and attack traffic is increasingly injected into the network. Specifically, we randomly select $a$% of total attack traffic, measured in both packet- and flow-level. As the value of $a$ increased, so does the attack intensity.

The same training dataset $T_r$, used in previous experiments, is also used in this experiment. The increase of the attack traffic is cumulatively achieved, whereby the increased attack traffic contains the previous attack traffic. The traffic injection procedure is repeated 10 times with different random seeds. We do not use separate symbols to differentiate packet- and flow-level datasets, since they are structurally the same. Hence, dataset used in this experiment is denoted as $D_{exp3} = d^a_{k,r}(I)$, where $1 \le r \le 10$, $a \in \{1\%, 5\%, 10\%, 25\%, 50\%, 75\%\}$ and $2 \le k \le 14$. In total, 780 test cases are used to assess the performance of each DDoS detection technique, for both packet- and flow-level attack traffic injections.

Similarly as in experiment 2, we use Pearson Correlation Coefficient test to quantitatively evaluate the impact of the attack intensities. Results of packet- and flow-level injections are shown in Tables 6 and 7, respectively. As expected, D-WARD presents a strong positive correlation with the increasing attack intensities in terms of accuracy and sensitivity. With the attack intensity increases, the amount of observed traffic increases as well, which is critical for D-WARD to detect attack traffic correctly.

Except for KMeans and the family of SVM techniques, all ML-based techniques present a weak or moderate correlation between the accuracy and the attack intensity. KMeans, SVM and Poly-SVM present a strong positive correlation between the increasing attack intensities and their accuracy. These techniques tend to label samples as attack traffic rather than legitimate. As attack intensities increase, the accuracy score is increased accordingly. RBF-SVM, contrarily, tends to label samples as legitimate traffic rather than attack. Hence, it presents a strong negative correlation with the increasing attack intensities. The correlation presented by sensitivity and specificity is not noticeably reflected in the score. In this experiment, the legitimate traffic remains the same through different attack intensities. In other words, the TN and FP are not changed with different attack intensities. Hence, the specificity score should be irrelevant to the changing of the attack intensity. Regarding the sensitivity score, since techniques exam the packet record one by one independently, the true positive rate of the same model should not be affected by attack intensities.

### 5.4. Exp. 4 – Impact of the class imbalance problem

The class imbalance problem is frequently encountered in practice, where the number of observations of one class is far less than the other class. When this problem occurs in the testing phase, accuracy alone is no longer enough to assess the performance of the detection scheme. Different types of evaluation metrics, such as sensitivity and specificity used in this work, need to be used to complement the accuracy to better assess performance. When the class imbalance problem occurs in the training dataset, it may hinder the learning process of classification algorithms [83]. Practically, if the imbalanced class distribution in the training dataset matches the native class prevalences in the test scenario, then the dataset bias in the learning process can be neglected. Yet, the described scenario does not apply to DDoS attack detection.

From the perspective of a DDoS attack detector, attack traffic usually represents a very small subset of all network traffic it observed, particularly in stealth attacks. However, when an attack happens, the attack traffic may become the majority class among the traffic that is observed by the detector. Hence, it is common that the detection scheme is dealing with highly imbalanced dataset, and the dominant class is non-stationary. Represented in our benchmark, the attack traffic is the minority class during the training phase, but it becomes the majority class after the attacker increases the attacking volume, shown in Fig. 6.

To assess the impact of the class imbalance problem in the training datasets, we generate a set of training data with different degrees of imbalance. We apply a simple random under-sampling method to create five subsets from the training dataset $T_r$. Each subset contains 70,000 packets, and the percentage of attack traffic in each subset is drawn from {10%, 30%, 50%, 70%, 90%}. All ML-based techniques are trained with each subset independently. Five models are then built for each technique. Due to the limited number of ICMP legitimate traffic, only TCP traffic is considered in this experiment. For testing, we apply five trained models on the testing dataset $T_s$ from Exp.1. The results show that the family of SVM techniques exhibits the strongest sensitivity to the imbalanced training datasets, while other techniques are affected slightly.

To further study the correlation between the class imbalance of the training data and the performance, we generate nine subsets from $T_s$ with different ratios of attack to legitimate traffic packets. Each subset contains 100,000 packets, and the percentage of attack traffic in each subset is drawn from $\{10\%, 20\%, 30\%, \ldots, 90\%\}$. The procedure of generating testing samples is repeated 10 times. In total, 90 test cases are used to assess the impact of the imbalanced and balanced training models on the performance of each technique. Figs. 14–16 display the results of linear SVM, Poly-SVM and RBF-SVM. Trained with small percentage of attack traffic, all SVM models tend to label most samples as legitimate traffic. Using a relatively balanced training dataset, the results show that the

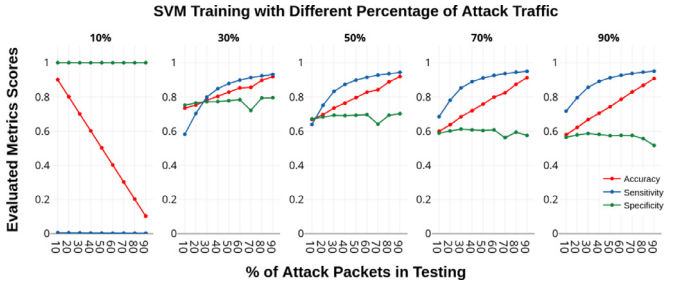


**Fig. 14.** Class Imbalance Problem Analysis - Linear SVM. The subtitle of each figure depicts the ratio of attack and legitimate traffic in the training sets. X-axis present the ratio of attack and legitimate traffic in the testing case.

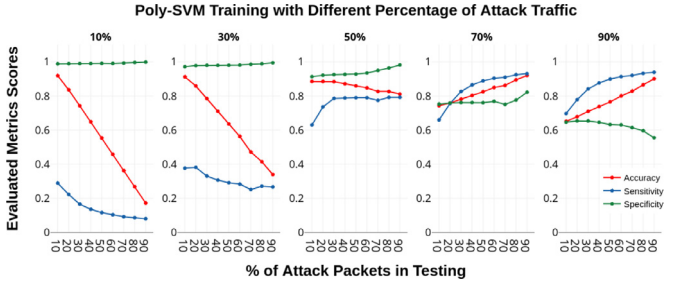


**Fig. 15.** Classimbalance problem Analysis - Poly-SVM.

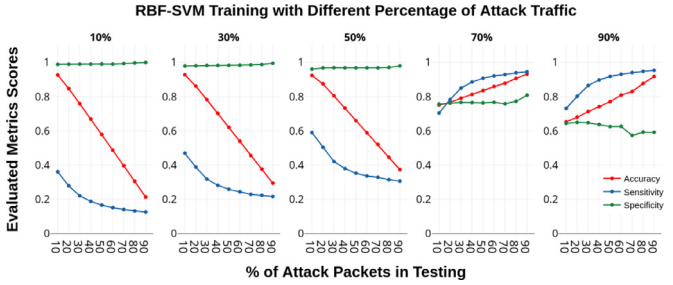


**Fig. 16.** Class imbalance problem analysis - RBF-SVM.

robustness of the model is improved. It is worth noting, however, that a balanced dataset does not necessarily achieve the best performance. A sophisticated kernel, such as RBF-SVM, can identify the hidden information by mapping features into higher dimensions. Specifically, RBF-SVM outperforms linear SVM when extremely imbalanced training data is used, although its overall performance remain less than optimal. On the other hand, the sophisticated kernel exhibits higher sensitivity to the data balancing, making it difficult to optimize its performance without degrading the robustness of the model.

In summary, the results clearly show that the impact of the class imbalance problem in datasets should not be neglected. Carefully designing the training process, analyzing the application scenario and choosing the appropriate method are critical for a successful intelligent DDoS detection scheme. Additionally, detection DDoS attacks in dynamically changing environment remains a challenge for ML-based detection methods.

## 6. Conclusion and future directions

In this work, we conducted a series of experiments to explore the advantages and limitations of DDoS detection techniques, and gain a better understanding of the influential factors that impact their performance. Rather than focusing on the intricacies of a particular solution, the focus of this paper is on empirically evaluating the "building blocks", which are commonly used in intelligent DDoS detection schemes. A comparative analysis of the overall performance of these techniques and a set of sensitivity analysis of the schemes to different influential factors are carried out, to provide a better understanding of the techniques' capabilities to detect DDoS attacks. We summarize the results as below:

- Although no single technique that outperforms all others in all test cases, the superiority of the detection capabilities exhibited by ML-based techniques is evident.
- Different techniques exhibit different preferences of feature types, emphasizing the significance of feature selection in developing intelligent DDoS detection strategies.
- The observed traffic proportions severely impact the performance of traditional DDoS detection methods that rely on monitoring the two-way traffic, while ML-based techniques display weak correlation with the proportion of the observed traffic.
- The impact of the class imbalance problem on the performance of ML-based techniques should not be underestimated, especially with respect to the dynamically evolving nature of DDoS attacks.

Finally, we provide a vision towards future research direction, based on our experiments' results.

- The majority of papers reviewed in this survey present their detection model as an isolated system and focus on how to improve the detection performance for a given dataset. To deploy the proposed ML-based detection techniques in real-world online detection systems, a number of challenges dealing with streaming data, such as buffering, overheads, traffic sampling, etc., have to be addressed.
- The different preferences over features by different techniques emphasize the critical impact of feature selections. Efforts should be expanded into an ensemble of intelligent schemes, strategically distributed across the network, using an appropriate feature selection model for efficient DDoS detection.
- On the other hand, over tailoring the feature selection may degrade the robustness of a detection model by over-fitting to a specific dataset, hence cause the failure in detecting unseen attack in the future. To build an effective DDoS defense system, the challenge of balancing trade-offs between detection in known dataset and unseen attacks has to be addressed.
- To address the above described dilemma, Deep Learning, which have been applied widely and successfully in pattern recognition, computer vision, natural language processing and etc., could have a novel application in distinguishing DDoS attacks. Deep learning methods can automatically extract complex representations hidden inside the raw input, in which the handcrafted feature selection is not necessary. The promising capabilities of learning complicated relationships presented by Deep Learning could also enhance the detection schemes' ability to reflect the "nature" of the attack.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Appendices

Tables 8 and 9 summarize schemes reviewed in this paper, along with the dataset used and evaluation results reported in each article.

**Table 8**
Reference summary.

| Categories | Techniques | Ref. | Year | Dataset Used | Performance |
|---|---|---|---|---|---|
| Classification | SVM | [30] | 2008 | local testbed<br>LLDoS dataset [84] | 92.8%–100% (Accuracy); 0–4.7(FPR); 0–1(FNR) |
| | | [31] | 2008 | local testbed | 93.38%(Accuracy); 0.018–0.029 (FPR); 0.006–0.0293(FNR) |
| | | [23] | 2009 | DARPA 1999 [76]<br>LLDoS dataset | 65.1%–100%(Detection Rate); 0–0.336(False Alarm Rate) |
| | | [24] | 2010 | LLDoS dataset | above 93.0%(Accuracy) |
| | | [25] | 2014 | LLDoS dataset | 95.11%(Accuracy); 0.008(FPR) |
| | | [26] | 2014 | darknet packets[1] | 95.25%(Best Detection Rate) |
| | | [27] | 2017 | DARPA 1999<br>CAIDA 2007 [75] | 96.55%(Accuracy) |
| | | [28] | 2017 | local testbed | 80%(Accuracy); 80%(Precision); 80%(Recall) |
| | | [29] | 2017 | local testbed | 100%(Detection Rate) |
| | ANN | [33] | 2009 | KDD Cup 1999 [85] | 79%–100%(Detection Rate); 0–0.15(FPR);0–0.06(FNR) |
| | | [41] | 2010 | local testbed | 99.45%–100%(Accuracy);0(FNR);0–0.0055(FPR) |
| | | [39] | 2011 | DARPA 1999<br>DARPA 2000 [84]<br>Conficker Dataset [86]<br>local testbed | 90.6%–99.4%(TPR);0.029–0.04(FPR) |
| | | [38] | 2012 | local testbed | −1.79–2.88(test error) |
| | | [34] | 2015 | local testbed | – |
| | | [35] | 2015 | DARPA 2000<br>LLDoS | 94%(Accuracy) |
| | | [36] | 2016 | local testbed | 92%–98%(Accuracy);0.88–0.95(Sensitivity);0.96–1(Specificity);0.96–1(Precision) |
| | | [40] | 2016 | CAIDA 2007<br>DAPRA 2009<br>local testbed | 96.3%–99%(Accuracy) |
| | | [42] | 2016 | darknet packets[1] | 77.8%–97.6%(F1 Measure) |
| | | [37] | 2016 | local testbed | 82.1%–95.6%(Accuracy) |
| | Decision Tree | [51] | 2008 | local testbed | 98.8%(Detection Rate);0.3%(FPR) |
| | | [48] | 2009 | local testbed | 89.4%–92.6%(Accuracy) |
| | | [45] | 2011 | local testbed | 2%–10%(FNR);1.5%–50%(False Classification Ratio) |
| | | [52] | 2012 | KDD Cup 1999<br>local testbed | 99.48%–99.97(Accuracy) |
| | | [49] | 2013 | Honeypot Traffic | 98.1%–98.3%(TP);0.01%–2.1%(FP);97.9%–99.9%(TN);1.7%–1.9%(FN) |
| | | [46] | 2014 | CAIDA 2007 | 0.98–0.99(Accuracy);0.77–0.99(Precision);0.98–1(Recall);0.99(Specificity) |
| | | [50] | 2014 | CAIDA | 0.996–0.999(Precision); 0.997–0.998(Recall) |
| | | [53] | 2016 | NSL-KDD | 99.67%(Accuracy) |
| | | [47] | 2016 | not specified | 0.986-0.993(TP); 0.001–0.006(FP) |

1. The dataset was collected by National Institute of Information and Communication Technology in Japan.

**Table 9**
Reference Summary – Continued

| Categories | Techniques | Ref. | Year | Dataset Used | Performance |
|---|---|---|---|---|---|
| Classification | Naive Bayes Classifier | [57] | 2008 | DARPA 2000 | – |
| | | [58] | 2008 | local testbed | 99.88%(Accuracy) |
| | | [55] | 2010 | KDD Cup 1999 | 99.2%-99.84%(Detection Rate); 0.03%-6.81%(FPR) |
| | | [60] | 2011 | SETS[2] | 97%-99.5%(Accuracy);<br>0.4%-7%(False Alarm Rate);<br>0–2.3%(Miss Rate) |
| | | [62] | 2012 | NSL-KDD | 65.4%-98.2%(Recall) |
| | | [64] | 2014 | KDD Cup 1999 | – |
| | | [61] | 2014 | 1998 World Cup Website Data | 95%(Detection Rate); 20% (FPR) |
| | | [56] | 2015 | local testbed | 96.61%-99.8%(Accuracy) |
| | | [59] | 2016 | local testbed<br>Conficker Dataset<br>ISOT Botnet Dataset[3] | 95.33%-99.6%(Accuracy);<br>97.5%-99.7%(Precision);<br>97.1%-94%(Recall) |
| | | [63] | 2016 | local testbed | 90.64%-95.93%(Accuracy) |
| Clustering | K-Means | [69] | 2007 | – | – |
| | | [70] | 2010 | local testbed | 96.53%-100%(Detection Rate);0%-2.54%(Error Rate) |
| | | [66] | 2010 | DARPA 2000 | – |
| | | [67] | 2011 | local testbed<br>honeypot traffic | 96.68%-100%(Correctly Classified); 0%-3%(Miss-Classified) |
| | | [68] | 2015 | local testbed | 67.72%-100%(Accuracy);<br>65%-100%(Detection Rate);<br>0-0.22(FPR) |

(*continued on next page*)

**Table 9** (*continued*)

| Categories | Techniques | Ref. | Year | Dataset Used | Performance |
|---|---|---|---|---|---|
| Nearest Neighbor Based | KNN | [71] | 2010 | DARPA 2000 | 91.886%(Correct Classification) 8.114%(Miss-Classified) |
| | | [72] | 2011 | local testbed | 97.42% & 78% (Best Accuracy) |
| | | [73] | 2015 | local testbed<br>KDD Cup 1999 | 88.4%-96.7%(Accuracy) |
| | | [74] | 2018 | LLDoS<br>KDD Cup 1999<br>CAIDA 2007<br>CAIDA 2008 | 0.8612-0.9929(Precision); 0.8645-0.9913(Recall) |
| Statistical | D-WARD | [21] | 2002 | local testbed | 0.003%-0.43%(Misclassification) |

2. SETS are 2-days UDP traffic data collected in India.
3. The dataset is from Univeristy of Victoria.

## References


[1] T. Ibragimov, et al., DDoS attacks in Q2 2018, (Webpage. [Online]. Available: https://securelist.com/ddos-report-inq2-2018/86537/).
[2] R. Hackett, Github triumphant over its Íargest everćyber pummeling, (Webpage. [Online]. Available:http://fortune.com/2015/04/03/github-ddos-china/).
[3] L.H. Newman, What we know about friday's massive east coast internet outage, (Webpage. [Online]. Available: https://www.wired.com/2016/10/internet-outageddos-dns-dyn/).
[4] M. Broersma, UK National Lottery Hit By Peak-Time DDoS Attack, (Webpage. [Online]. Available: https://www.silicon.co.uk/security/uk-national-lotteryddos-222601?inf_by=5b7ee807671db80d6d8b4982).
[5] S. Kottler, Github – February 28th DDoS Incident Report, 2018, ([Online]. Available: https://githubengineering.com/ddos-incident-report/).
[6] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J.A. Halderman, L. Invernizzi, M. Kallitsis, et al., Understanding the mirai botnet, USENIX Security Symposium, 2017.
[7] J. Mirkovic, P. Reiher, A taxonomy of DDoS attack and DDoS defense mechanisms, SIGCOMM Comput. Commun. Rev. 34 (2) (2004) 39–53.
[8] T. Peng, C. Leckie, K. Ramamohanarao, Survey of network-based defense mechanisms countering the DoS and DDoS problems, ACM Comput. Surv. (CSUR) 39 (1) (2007) 3.
[9] H. Beitollahi, G. Deconinck, Analyzing well-known countermeasures against distributed denial of service attacks, Comput. Commun. 35 (11) (2012) 1312–1332.
[10] S.T. Zargar, J. Joshi, D. Tipper, A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks, IEEE Commun. Surv. Tutor. 15 (4) (2013) 2046–2069.
[11] M.H. Bhuyan, H.J. Kashyap, D.K. Bhattacharyya, J.K. Kalita, Detecting distributed denial of service attacks: methods, tools and future directions, Comput. J. 57 (4) (2013) 537–556.
[12] O. Osanaiye, et al., Distributed denial of service resilience in cloud: review and conceptual cloud DDoS mitigation framework, J. Netw. Comput. Appl. 67 (2016) 147–165.
[13] G. Somani, M.S. Gaur, D. Sanghi, M. Conti, R. Buyya, DDoS attacks in cloud computing: Issues, taxonomy, and future directions, Comput. Commun. 107 (2017) 30–48.
[14] Q. Yan, F.R. Yu, Q. Gong, J. Li, Software-defined networking (SDN) and distributed denial of service attacks in cloud computing environments: a survey, some research issues, and challenges, IEEE Commun. Surv. Tutor. 18 (1) (2016) 602–622.
[15] M. Roesch, et al., Snort: Lightweight intrusion detection for networks., in: Lisa, 99, 1999, pp. 229–238.
[16] M. Ahmed, A.N. Mahmood, J. Hu, A survey of network anomaly detection techniques, J. Netw. Comput. Appl. 60 (2016) 19–31.
[17] S.X. Wu, W. Banzhaf, The use of computational intelligence in intrusion detection systems: a review, Appl. Soft Comput. 10 (1) (2010) 1–35.
[18] M.H. Bhuyan, et al., Network anomaly detection: methods, systems and tools, IEEE Commun. Surv. Tutor. 16 (1) (2014) 303–336.
[19] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, et al., Anomaly-based network intrusion detection: techniques, systems and challenges, Comput. Secur. 28 (1–2) (2009) 18–28.
[20] A.L. Buczak, E. Guven, A survey of data mining and machine learning methods for cyber security intrusion detection, IEEE Commun. Surv. Tutor. 18 (2) (2016) 1153–1176.
[21] J. Mirkovic, G. Prier, P. Reiher, Attacking DDoS at the source, in: International Conference on Network Protocols, IEEE, 2002, pp. 312–321.
[22] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[23] J. Cheng, J. Yin, Y. Liu, Z. Cai, C. Wu, DDoS attack detection using IP address feature interaction, in: International Conference on Intelligent Networking and Collaborative Systems, IEEE, 2009, pp. 113–118.
[24] Y. Liu, J. Yin, J. Cheng, B. Zhang, Detecting DDoS attacks using conditional entropy, in: International Conference on Computer Application and System Modeling, 13, IEEE, 2010, pp. V13–278.
[25] R. Kokila, S.T. Selvi, K. Govindarajan, DDoS detection and analysis in SDN-based environment using support vector machine classifier, in: International Conference on Advanced Computing, IEEE, 2014, pp. 205–210.
[26] N. Furutani, T. Ban, J. Nakazato, J. Shimamura, J. Kitazono, S. Ozawa, Detection of DDoS backscatter based on traffic features of darknet TCP packets, in: Asia Joint Conference on Information Security, IEEE, 2014, pp. 39–43.
[27] J. Liu, Y. Lai, S. Zhang, FL-GUARD: A detection and defense system for DDoS attack in SDN, in: International Conference on Cryptography, Security and Privacy, ACM, 2017, pp. 107–111.
[28] N. Meti, D. Narayan, V. Baligar, Detection of distributed denial of service attacks using machine learning algorithms in software defined networks, in: International Conference on Advances in Computing, Communications and Informatics, IEEE, 2017, pp. 1366–1371.
[29] D. Hu, P. Hong, Y. Chen, FADM: DDoS flooding attack detection and mitigation system in software-defined networking, in: IEEE Global Communications Conference, IEEE, 2017, pp. 1–7.
[30] T. Xu, D. He, Y. Luo, DDoS attack detection based on RLT features, in: International Conference on Computational Intelligence and Security, IEEE, 2007, pp. 697–701.
[31] J. Yu, H. Lee, M.-S. Kim, D. Park, Traffic flooding attack detection with SNMP MIB using SVM, Comput. Commun. 31 (17) (2008) 4212–4219.
[32] B. Yegnanarayana, Artificial neural networks, PHI Learning Pvt. Ltd., 2009.
[33] I. Ahmad, A.B. Abdullah, A.S. Alghamdi, Application of artificial neural network in detection of DoS attacks, in: International conference on Security of information and networks, ACM, 2009, pp. 229–234.
[34] T. Zhao, D.C.-T. Lo, K. Qian, A neural-network based DDoS detection system using hadoop and hbase, in: International Symposium on Cyberspace Safety and Security, 2015 IEEE 12th International Conference on High Performance Computing and Communications, 2015, pp. 1326–1331.
[35] C.-J. Hsieh, T.-Y. Chan, Detection DDoS attacks based on neural-network using apache spark, in: International Conference on Applied System Innovation, IEEE, 2016, pp. 1–4.
[36] A. Saied, R.E. Overill, T. Radzik, Detection of known and unknown DDoS attacks using artificial neural networks, Neurocomputing 172 (2016) 385–393.
[37] D. Peraković, M. Periša, I. Cvitić, S. Husnjak, Artificial neuron network implementation in detection and classification of ddos traffic, in: Telecommunications Forum, IEEE, 2016, pp. 1–4.
[38] B.B. Gupta, R.C. Joshi, M. Misra, ANN based scheme to predict number of zombies in a DDoS attack., IJ Netw. Secur. 14 (2) (2012) 61–70.
[39] P.A.R. Kumar, S. Selvakumar, Distributed denial of service attack detection using an ensemble of neural classifier, Comput. Commun. 34 (11) (2011) 1328–1341.
[40] K. Johnson Singh, K. Thongam, T. De, Entropy-based application layer DDoS attack detection using Artificial Neural Networks, Entropy 18 (10) (2016) 350.
[41] J. Li, Y. Liu, L. Gu, DDoS attack detection based on neural network, in: International Symposium on Aware Computing, IEEE, 2010, pp. 196–199.
[42] S.H.A. Ali, S. Ozawa, T. Ban, J. Nakazato, J. Shimamura, A neural network model for detecting DDoS attacks using darknet traffic features, in: International Joint Conference on Neural Networks, IEEE, 2016, pp. 2979–2985.
[43] B.D. Ripley, Pattern Recognition and Neural Networks, Cambridge university press, 2007.
[44] S.K. Murthy, Automatic construction of decision trees from data: a multi-disciplinary survey, Data Min. Knowl. Discov. 2 (4) (1998) 345–389.
[45] Y.-C. Wu, H.-R. Tseng, W. Yang, R.-H. Jan, DDoS detection and traceback with decision tree and grey relational analysis, Int. J. Ad Hoc Ubiquit.Comput. 7 (2) (2011) 121–136.
[46] E. Balkanli, J. Alves, A.N. Zincir-Heywood, Supervised learning to detect DDoS attacks, in: IEEE Symposium on Computational Intelligence in Cyber Security, IEEE, 2014, pp. 1–8.
[47] A. Degirmencioglu, H.T. Erdogan, M.A. Mizani, O. Yılmaz, A classification approach for adaptive mitigation of SYN flood attacks: preventing performance loss due to syn flood attacks, in: Network Operations and Management Symposium, IEEE, 2016, pp. 1109–1112.
[48] W. Lu, M. Tavallaee, G. Rammidi, A.A. Ghorbani, BotCop: An online botnet traffic classifier, in: Communication Networks and Services Research Conference, IEEE, 2009, pp. 70–77.

[49] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, D. Garant, Botnet detection based on traffic behavior analysis and flow intervals, Comput. Secur. 39 (2013) 2–16.
[50] K. Singh, S.C. Guntuku, A. Thakur, C. Hota, Big data analytics framework for peer-to-peer botnet detection using random forests, Inf. Sci. 278 (2014) 488–497.
[51] W. Wang, S. Gombault, Efficient detection of DDoS attacks with important attributes, in: International Conference on Risks and Security of Internet and Systems, IEEE, 2008, pp. 61–67.
[52] H.J. Kashyap, D. Bhattacharyya, A DDoS attack detection mechanism based on protocol specific traffic features, in: International Conference on Computational Science, Engineering and Information Technology, ACM, 2012, pp. 194–200.
[53] O. Osanaiye, H. Cai, K.-K.R. Choo, A. Dehghantanha, Z. Xu, M. Dlodlo, Ensemble-based multi-filter feature selection method for ddos detection in cloud computing, EURASIP J. Wirel. Commun.Netw. 2016 (1) (2016) 130.
[54] K.P. Murphy, Naive Bayes Classifiers, University of British Columbia 18 (2006).
[55] D.M. Farid, N. Harbi, M.Z. Rahman, Combining Naive Bayes and decision tree for adaptive intrusion detection, arXiv preprint arXiv:1005.4496 (2010).
[56] V. Katkar, A. Zinjade, S. Dalvi, T. Bafna, R. Mahajan, Detection of dos/ddos attack against http servers using naive Bayesian, in: International Conference on Computing Communication Control and Automation, IEEE, 2015, pp. 280–285.
[57] S. Benferhat, T. Kenaza, A. Mokhtari, A Naive Bayes approach for detecting coordinated attacks, in: International Computer Software and Applications, IEEE, 2008, pp. 704–709.
[58] S. Noh, G. Jung, K. Choi, C. Lee, Compiling network traffic into rules using soft computing methods for the detection of flooding attacks, Appl. Soft Comput. 8 (3) (2008) 1200–1210.
[59] G. Kirubavathi, R. Anitha, Botnet detection via mining of traffic flow characteristics, Comput. Electr. Eng. 50 (2016) 91–101.
[60] R. Vijayasarathy, S.V. Raghavan, B. Ravindran, A system approach to network modeling for DDoS detection using a Naive Bayesian classifier, in: International Conference on Communication Systems and Networks, IEEE, 2011, pp. 1–10.
[61] S. Umarani, D. Sharmila, Predicting application layer ddos attacks using machine learning algorithms, Int. J. Comput.Control Quant.Inf. Eng. 8 (10) (2014).
[62] N. Sharma, S. Mukherjee, Layered approach for intrusion detection using Naïve Bayes classifier, in: International Conference on Advances in Computing, Communications and Informatics, ACM, 2012, pp. 639–644.
[63] R.F. Fouladi, C.E. Kayatas, E. Anarim, Frequency based DDoS attack detection approach using naive bayes classification, in: International Conference on Telecommunications and Signal Processing, IEEE, 2016, pp. 104–107.
[64] S. Veetil, Q. Gao, Real-time network intrusion detection using hadoop-based Bayesian classifier, in: Emerging Trends in ICT Security, Elsevier, 2014, pp. 281–299.
[65] C.C. Aggarwal, C.K. Reddy, Data Clustering: Algorithms and Applications, CRC press, 2013.
[66] L. Zi, J. Yearwood, X.-W. Wu, Adaptive clustering with feature ranking for DDoS attacks detection, in: International Conference on Network and System Security, IEEE, 2010, pp. 281–286.
[67] W. Lu, G. Rammidi, A.A. Ghorbani, Clustering botnet communication traffic based on n-gram feature selection, Comput. Commun. 34 (3) (2011) 502–514.
[68] Q. Liao, H. Li, S. Kang, C. Liu, Application layer DDoS attack detection using cluster with label based on sparse vector decomposition and rhythm matching, Secur. Commun. Netw. 8 (17) (2015) 3111–3120.
[69] J. Yu, Z. Li, H. Chen, X. Chen, A detection and offense mechanism to defend against application layer DDoS attacks, in: International Conference on Networking and Services, IEEE, 2007, p. 54.
[70] R. Zhong, G. Yue, DDoS detection system based on data mining, in: International Symposium on Networking and Network Security, 2010, pp. 2–4.
[71] H.-V. Nguyen, Y. Choi, Proactive detection of DDoS attacks utilizing k-NN classifier in an anti-DDoS framework, Int. J. Electr. Comput. Syst. Eng. 4 (4) (2010) 247–252.
[72] M.-Y. Su, Real-time anomaly detection systems for Denial-of-Service attacks by weighted k-nearest-neighbor classifiers, Expert Syst. Appl. 38 (4) (2011) 3492–3498.
[73] P. Xiao, W. Qu, H. Qi, Z. Li, Detecting DDoS attacks against data center with correlation analysis, Comput. Commun. 67 (2015) 66–74.
[74] L. Zhu, X. Tang, M. Shen, X. Du, M. Guizani, Privacy-preserving DDoS attack detection using cross-domain traffic in Software Defined Networks, IEEE J. Sel. Areas Commun. 36 (3) (2018) 628–643.
[75] CAIDA, The CAIDA 'DDoS Attack 2007' Dataset, (Webpage. [Online]. Available: https://www.caida.org/data/passive/ddos-20070804_dataset.xml).
[76] M.I.e. Lincoln Laboratory, 1999 DARPA Intrusion Detection Scenario Specific Datasets, (Webpage. [Online]. Available: https://www.ll.mit.edu/ideval/data/1999data.html).
[77] Tcpreplay, Tcpreplay, (Webpage. [Online]. Available: http://tcpreplay.synfin.net/).
[78] D.M. Powers, Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation, J. Mach. Learn. Technol. 2 (1) (2011) 37–63.
[79] A.P. Bradley, The use of the area under the roc curve in the evaluation of machine learning algorithms, Pattern Recognit. 30 (7) (1997) 1145–1159.
[80] TCPDUMP, Tcpdump, (Webpage. [Online]. Available: www.tcpdump.org).
[81] A. Orebaugh, G. Ramirez, J. Beale, Wireshark & Ethereal Network Protocol Analyzer Toolkit, Elsevier, 2006.
[82] B. Claise, Cisco systems netflow services export version 9 (2004).
[83] B. Krawczyk, Learning from imbalanced data: open challenges and future directions, Prog. Artif. Intell. 5 (4) (2016) 221–232.
[84] M.I.T. Lincoln Laboratory, 2000 Intrusion Detection Scenario Specific Datasets, (Webpage. [Online]. Available: https://www.ll.mit.edu/r-d/datasets/2000-darpaintrusion-detection-scenario-specific-datasets).
[85] KDD cup 1999 data, (Webpage. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html).
[86] Center for Applied Internet Data Analysis, UCSD Network Telescope – Three Days of Conficker Dataset, (Webpage. [Online]. Available: http://www.caida.org/data/passive/telescope-3daysconficker_dataset.xml).

**Xiaoyu Liang** is a Ph.D. student at the department of Computer Science at University of Pittsburgh. She received her MS degree from Ohio University (2012). Her current research interests focus on network security and Artificial Intelligent. Her dissertation research involves designing and building a distributed defense mechanism to detect and mitigate Distributed Denial of Service attacks.

**Dr. Znati** is a Professor in the Department of Computer Science, with a joint appointment in Computer Engineering at the School of Engineering. Dr. Znati served as the Director of the Computer and Network Systems Division at the National Science Foundation. Prior to this assignment, he also served as a Senior Program Director for Networking Research Program at NSF. In addition to his NR responsibilities, Dr. Znati led the Information Technology Research (ITR) Initiative, a cross-directorate research program, and served as the Committee Chair of the NSF Information Technology Research Initiative. Dr. Znati's main research interests are in the design and analysis of evolvable, secure and resilient network architectures and protocols for wired and wireless communication networks, the design of new fault tolerant mechanisms for energy-aware resiliency in data-intensive computing, and exploring new approaches to enable and facilitate model reuse to enable effective computational modeling and simulation frameworks to guide the design of engineered complex networked systems. Dr. Znati has served as the General Chair of several main conferences, including GlobeCom 2010, IEEE INFOCOM 2005, SECON 2004, the first IEEE conference on Sensor and Ad Hoc Communications and Networks, the Annual Simulation Symposium, and the Communication Networks and Distributed Systems Modeling and Simulation Conference. He also served or currently serves as a member of Editorial Boards of a number of networking, distributed system and security journals and transactions.