SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

May 2023, Vol. 66 152105:1–152105:16
https://doi.org/10.1007/s11432-021-3545-0

# Towards real-time ML-based DDoS detection via cost-efficient window-based feature extraction

Haibin LI[1], Yi ZHAO[1*], Wenbing YAO[1], Ke XU[1,3*] & Qi LI[2,3]

[1]*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084*, China;*
[2]*Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing* 100084*, China;*
[3]*Beijing National Research Center for Information Science and Technology (BNRist), Beijing* 100084*, China*

Received 15 December 2021/Revised 20 March 2022/Accepted 4 July 2022/Published online 17 April 2023

**Abstract** Distributed denial of service (DDoS) detection is still an open and challenging problem. In particular, sophisticated attacks, e.g., attacks that disguise attack packets as benign traffic always appear, which can easily evade traditional signature-based methods. Due to the low requirements for computing resources compared to deep learning, many machine learning (ML)-based methods have been realistically deployed to address this issue. However, most existing ML-based DDoS detection methods are highly dependent on the features extracted from each flow, which incur remarkable detection delay and computation overhead. This article investigates the limitations of typical ML-based DDoS detection methods caused by the extraction of flow-level features. Moreover, we develop a cost-efficient window-based method that extracts features from a fixed number of packets periodically, instead of per flow, aiming to reduce the detection delay and computation overhead. The newly proposed window-based method has the advantages of well-controlled overhead and wide support of common routers due to its simplicity and high efficiency by design. Through extensive experiments on real datasets, we evaluate the performance of flow-based and window-based methods. The experimental results demonstrate that our proposed window-based method can significantly reduce the detection delay and computation overhead while ensuring detection accuracy.

**Keywords** DDoS attack, machine learning, feature extraction, detection delay, cost-efficiency

**Citation** Li H B, Zhao Y, Yao W B, et al. Towards real-time ML-based DDoS detection via cost-efficient window-based feature extraction. Sci China Inf Sci, 2023, 66(5): 152105, https://doi.org/10.1007/s11432-021-3545-0

## 1 Introduction

The Internet, delivering vast information and other resources, has been changing the world. Services and applications deployed on the Internet, such as email, banking, and e-commerce, significantly facilitate our daily life. Meanwhile, the Internet suffers serious security problems, such as various network attacks [1]. In particular, distributed denial of service (DDoS) attacks exhaust resources and disrupt services on the Internet with a large volume of packets. To avoid being detected, some DDoS attackers attempt to mimic benign traffic, making it challenging to detect them [2, 3]. Traditional DDoS detection methods cannot effectively detect those new and complex attacks.

Traditional detection methods, in most cases, focus on a specific type of attack, for example, transmission control protocol (TCP) SYN flood or domain name system (DNS) amplification attack [4–6]. Despite high detection accuracy on a single attack form, they fail to capture other attacks with different attack patterns. As an emerging trend, machine learning (ML) has been applied to detect DDoS attacks [7–9]. ML algorithms are able to automatically identify attacks by leveraging knowledge learned from data of attack traffic, which greatly improves the detection performance. Thus, ML-based techniques for DDoS protection have also been adopted by industry [10]. Note that the referred ML-based techniques in this paper refer to ML-based methods that rely on feature extraction, rather than deep learning (DL)-based methods that can automatically learn features from the data instead of handcrafted feature extraction. In fact, DL models have high requirements for equipment, such as high-performance computing and

* Corresponding author (email: zhao_yi@tsinghua.edu.cn, xuke@tsinghua.edu.cn)

 

large-scale data support, which makes it difficult for current network equipment to achieve realistic deployment of DL models. Compared to DL models, ML models usually have lower complexity and lower requirements for computing resources, which enables the chance to deploy ML models on network devices for DDoS detection.

However, the deployment of ML-based DDoS detection on network devices still encounters the following challenges. First, the computing resources of network devices are usually limited for running ML models, given that the devices have to ensure enough resources for the normal operation of their original network functions at the same time. Thus, the resource consumption of running ML-based DDoS detection models should be kept as low as possible, which is a challenging requirement. Second, reducing resource consumption of ML-based DDoS detection while maintaining the compatibility with different ML algorithms is a challenging task. It is inevitable that when one ML algorithm cannot detect DDoS attacks with satisfactory accuracy, other ML algorithms should be used to replace it. Thus, the design of a cost-efficient approach cannot be achieved by simply optimizing the efficiency of some specific ML algorithms. The cost-efficient approach must be carefully designed to reduce the detection cost while not affecting the use of arbitrary ML algorithms. Third, maintaining the overall accuracy performance of ML models with reduced resource consumption is also a challenging problem.

In this article, we systemically investigate existing ML-based DDoS attack detection methods. Because big DL-based models cannot easily fit into common network devices with constrained computing resources, most of the deployed common intelligent methods in the real world tend to utilize light-weight and easy-to-deploy ML-based algorithms, such as decision tree (DT) [11], random forest (RF) [12], and naive Bayes (NB) [13]. For machine learning, feature engineering is an important step because it determines whether the models can be trained on significant and effective information. Based on our investigation, we find that many ML-based DDoS detection methods extract flow-level features in feature engineering. To extract flow-level features, a typical procedure is to first hash (or aggregate) each packet into a corresponding flow defined by the 5-tuple in the packet header, i.e., protocol type, IP source address, source port number, IP destination address, and destination port number. Then, it computes feature values from each flow. Although feature extraction by flow-based method keeps detailed flow information, it incurs remarkable detection delay and computation overhead on the detection device due to the complex flow-tracking procedure. Furthermore, common network devices such as a gateway or home routers, have limited memory and computing resources. While deploying ML-based attack detection models on such devices, the run-time computation overhead of the ML models must be carefully controlled at a low level, so that the devices are able to run the ML model to detect attacks even when processing high-bandwidth traffic at the same time (e.g., under DDoS). To scale to high bandwidth scenarios, the delay of feature extraction for ML models should also be kept low in order to achieve real-time detection of attacks. Motivated by the aforementioned considerations, we focus on the following question: is it possible to further reduce the detection delay and computation overhead without using flow-level features, while achieving ensured detection accuracy?

To answer the question, we develop a novel window-based (a.k.a., win-based) feature extraction method that effectively reduces detection delay and computation overhead, while detecting DDoS attacks effectively with the same existing ML algorithms. To better deploy ML-based DDoS detection methods to resource constrained network devices, we aim to make the proposed method simple for deployment and effective for attack detection, with the advantages of well-controlled overhead, wide support of common routers, and accuracy preserving. By leveraging information within an interval (i.e., a detection window), the proposed method feeds the ML model with features extracted from a fixed number of packets among all flows periodically, instead of calculating features of individual flows, thus avoiding the detection delay and computation overhead of the flow-based method. To validate whether the proposed win-based method can achieve higher detection efficiency while ensuring detection accuracy, we conduct extensive experiments using multiple real DDoS attack datasets in offline and online (i.e., real-time) modes. Our experiments evaluate the performance of multiple typical ML-based DDoS methods with win-based and flow-based features. The comprehensive comparison results demonstrate the overall effectiveness of our proposed win-based feature extraction method in terms of accuracy and efficiency. Moreover, we discuss research directions of ML-based DDoS detection.

The contributions we make in this article are as follows:

• We systematically investigate typical ML-based DDoS detection schemes using flow-level features, and summarize detection delay and computation overhead problem of flow-based methods. We make meaningful exploration on the trade-off between cost and accuracy for ML-based DDoS detection.

• We propose a novel window-based feature extraction method for real-time ML-based DDoS detection. In addition to effectively reducing detection delay and computation overhead, the proposed win-based method is fully compatible with various ML-based methods, showing its capability of addressing the challenges of deploying ML-based DDoS detection models.

• Extensive experiments on real datasets demonstrate that our win-based method achieves comprehensive advantages in cost-efficiency and accuracy retention, as well as broad compatibility, which can further promote the deployment of ML methods in existing resource-constrained network equipments in the real world.

## 2 Background

In this section, we briefly introduce the problem of DDoS attack and detection. In particular, we elaborate on the methodology of typical ML-based DDoS detection by investigating existing studies, and summarize our observations.

### 2.1 DDoS attack and detection

DDoS attacks can be classified as semantic and brute-force ones [14]. Semantic attacks exploit the vulnerabilities, and bugs of protocol or software implemented at the victim side to consume resources. TCP SYN flood attack is an example of semantic attacks, in which the attacker exploits the weakness in the TCP three-way handshake to fill up the connection queue of the victim. Brute-force attacks flood the victims' machines with a tremendous number of legitimate packets that exhaust targeted resources. For example, a large number of zombie computers in a botnet send seemingly legitimate DNS requests simultaneously to a victim to overwhelm its resources. For semantic attacks, they can possibly be mitigated by modifying the vulnerable protocols or fixing the bugs. While for brute-force ones, there might be no such vulnerabilities or bugs to fix for mitigation, and additional mitigation strategies are required for defense. Note that semantic attacks and brute-force attacks may share some similarities, e.g., a large volume of traffic, and excess consumption of resources. Both semantic and brute-force attacks aim to make services unavailable by disrupting them or impeding legitimate requests. Attackers usually conduct attacks by manipulating bots to flood a target, in which cases responses to benign requests are disturbed or even suspended.

Existing DDoS detection methods can be categorized as either signature-based or anomaly-based methods. Signature-based methods are able to capture known attacks with high detection accuracy but are unable to work on unknown attacks. Anomaly-based methods analyze malicious or uncommon patterns of behavior by applying statistical and machine learning techniques to a large amount of data, rather than simply identifying particular signatures. They are effective in learning common patterns and capable of detecting unknown or suspicious behavior according to the features extracted from existing data, which differs from traditional methods that rely heavily on the domain knowledge of security experts.

### 2.2 ML-based DDoS detection

With regard to ML-based DDoS detection methods, there are usually three phases, namely, preprocessing phase, the training and validation phase, and the testing phase. The first phase prepares training data, the second adjusts parameters of the ML model, and the last checks the model's validity. These three phases are illustrated in Figure 1.

The preprocessing phase includes the following procedures: data collection that collects and labels data, feature extraction that extracts features from collected data, and feature selection that selects key features for detection.

• Data collection: Data (i.e., network traffic) in networks are captured at different locations like routers and victim servers [15]. Traffic sniffers can be deployed according to the network topology. Many tools such as pcap and wireshark, are used to capture packets.

• Feature extraction: Feature extraction relies on the knowledge of security experts, and the extracted features used in the detection can directly affect the accuracy. In general, most of the ML-based DDoS detection schemes extract features based on flows, such as [16–19]. For example, Wang et al. [18] extracted the same 41 features as the KDD Cup 99 dataset. Typical flow-based features include the number of packets per second, byte rate, and average packet size. Symbolic features are usually converted to

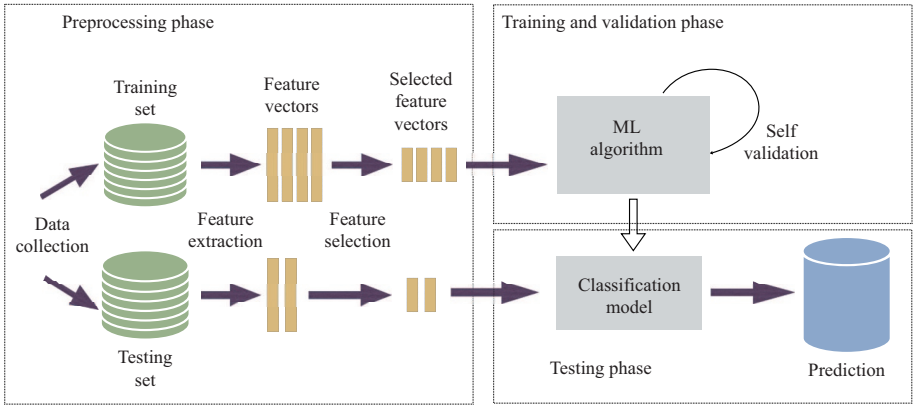


**Figure 1** (Color online) Workflow of typical ML-based DDoS detection.

numeric forms, and then normalized into the same range in order to guarantee unbiasedness. The feature extraction delay is significant, especially for data containing a large number of flows.

• Feature selection: Feature selection aims to reduce computation overhead, memory consumption, as well as the impact of feature noises. There are several approaches to selecting features. Barati et al. [17] introduced a genetic algorithm (GA), and tested combinations of flow-based features by the GA fitness function. In their experiments, they selected the five most efficient features out of the original 43 flow-based features based on classification accuracy. Osanaiye et al. [16] analyzed four feature selection methods and tested them with three machine learning algorithms.

The training and validation phase adapts the ML model's parameters and chooses hyper-parameters such as the number of trees in RF and the maximum depth of DT through a validation set randomly picked from training sets. Training and detection delay varies with different ML algorithms. DT, NB, and Bayes network (BN) usually have less delay than support vector machine (SVM), and ensemble learning methods. The testing phase utilizes the model trained in the previous phase to do classification or prediction on the testing samples containing features extracted from the testing set.

### 2.3 Detection delay and overhead

Detection delay and overhead are two important problems in online ML-based DDoS detection. In this paper, we use detection delay (or detection latency) to refer to the duration from the time when packets are collected to the time when the detection result is given. Detection overhead refers to the computing resources (e.g., CPU and memory) consumed by the detection procedure. Reducing detection delay and overhead is crucial for ML-based DDoS detection, especially when deploying the ML models in resource-constrained network devices.

Optimizing feature extraction is vital for addressing the problem of detection delay and overhead. Once the ML model is trained, it can be deployed at the detection point (e.g., network devices or destination servers) for online detection. During the process of online detection, the collected packets will first go through the preprocessing phase for feature extraction, and then go through the testing phase for model inference in order to do classification or prediction. Both the feature extraction and model inference can lead to detection delay and overhead. Optimizing the detection efficiency can be achieved by either improving the efficiency of feature extraction or improving the efficiency of model inference. However, improving the efficiency of model inference needs to optimize the specific ML algorithm, and the optimization will lose effectiveness when a new algorithm is needed to replace the old one. An alternative is to optimize the efficiency of feature extraction, which is the focus of our work.

We investigate the commonly used flow-based feature extraction method and find that it still needs further improvement. The flow-based method is a classic method for feature extraction. A flow is identified by 5-tuple, which contains the key information of the network connection establishment between any hosts. Analyzing network traffic based on 5-tuple is intuitive and has become a tradition. Many existing ML-based DDoS schemes use flow-based features, in which the packets have to be aggregated into flows before feature extraction. For example, Ref. [19] used the following flow-level features: average of packets per flow (APf), average of Bytes per flow (ABf), average of duration per flow (ADf), etc. Calculating flow-level features requires tracking flow identities and computing feature values for every

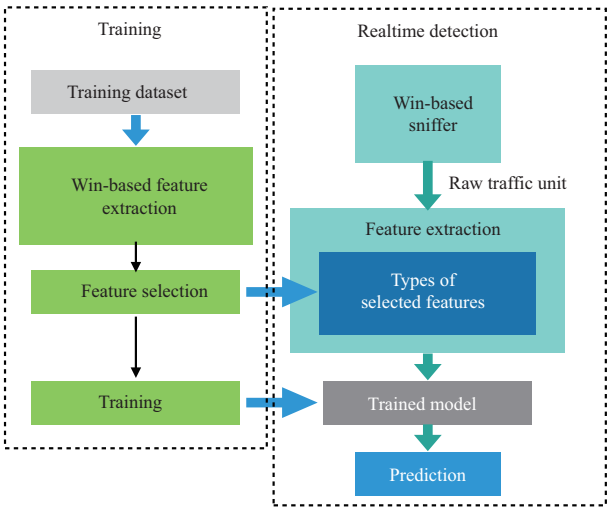


**Figure 2** (Color online) Our proposed ML-based realtime DDoS detection framework.

single flow. However, the number of flows passing a single network device could be very large, whether in normal traffic or DDoS traffic, which leads to non-negligible overhead for calculating flow-level features, and consequently causes non-negligible detection delay and overhead. What's worse, in DDoS scenarios, numerous zombie computers in a botnet may send packets simultaneously towards the victim, resulting in an explosion of flows on the network devices near the victim side. In that case, the calculation of flow-level features for numerous flows will cause a heavy burden on the DDoS detection devices, where the flow-based feature extraction might become the bottleneck of ML-based DDoS detection. In addition, when deploying ML-based DDoS models on resource-constrained network devices, we have to make sure that the running of ML models will not overwhelm the resources for maintaining the normal network function of the network device. Thus, a cost-efficient feature extraction approach is crucial for deploying ML-based DDoS detection models, while the flow-based feature extraction cannot fully meet the requirements of the task.

In summary, by investigating typical existing ML-based DDoS detection methods, we have the following key observations. ML-based methods are becoming a new trend in DDoS detection, showing the capability of ML-based methods for detecting DDoS attacks. However, the traditional flow-based feature extraction method still incurs significant detection delay and overhead, which is not conducive to the rapid detection of attacks and the stable operation of the detection device (e.g., routers) especially under the heavy pressure of DDoS. It is necessary to reduce latency such that ML methods can achieve better detection efficiency while maintaining detection effectiveness.

## 3 Reducing delay and overhead with win-based feature extraction

In this section, we explain the details of the proposed win-based feature extraction method which aims for reducing detection delay and computation overhead. We first introduce our proposed ML-based realtime DDoS detection framework, then elaborate on the design of the win-based method, and finally summarize the core characteristics of the proposed method.

Most existing ML-based DDoS detection methods are highly dependent on the features extracted from flows, which incurs significant detection delay and computation overhead. The root cause of the detection latency is that the commonly adopted flow-based feature extraction requires tracking flows, which is very time-consuming. To address the issue and reduce the detection overhead, we propose a new feature extraction method that extracts features according to a constant number of packets without respect to flow types and traffic situations. The intuition behind it is that DDoS packets share similarities without regard to the flow types, and ML models can be trained to distinguish whether a bunch of continuous packets in the packet stream contain attacks, making win-based DDoS detection practical.

We design an ML-based real-time DDoS detection framework shown in Figure 2. The training module is responsible for constantly learning patterns from the training data, and the real-time detection module is responsible for dynamically detecting attacks from the real-time network traffic. Note that both the

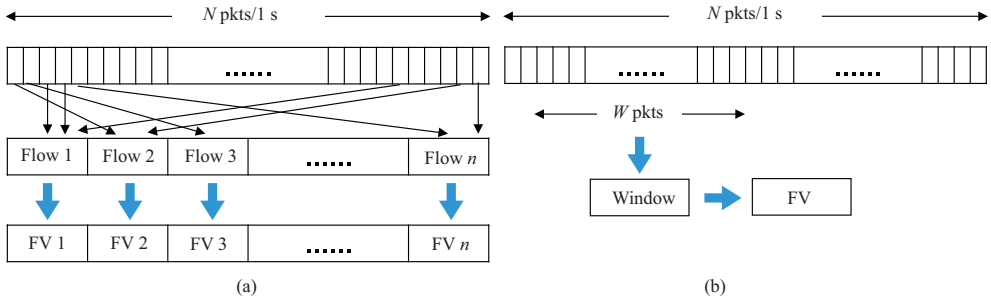


**Figure 3** (Color online) Flow-based (a) and win-based (b) methods. The win-based method does not distinguish different flows while the flow-based one does.

trained model and types of features in the real-time detection module can be dynamically updated. The key difference between our framework and traditional approaches is that we use a win-based feature extraction method in both training and real-time detection. Especially, to make the detection fast and scalable, we design a win-based sniffer which captures a fixed number of packets in a preset time window.

Our proposed win-based feature extraction method is illustrated in Figure 3(b), and the flow-based method is shown in Figure 3(a). In the win-based method, a fixed number ($W$, i.e., window size) of continuous packets are periodically (e.g., every one second) gathered and analyzed. For each period, all $W$ gathered packets in the window are treated as a unit that is converted into one feature vector, regardless of which flow each packet belongs to. Specifically, compared with the flow-based method, our proposed win-based method only generates one feature vector from $W$ continuously captured packets for each period, instead of assembling all $N$ packets into flows and extracting a range of feature vectors for all flows for the period. Obviously, with reduced complexity, the win-based method is effective in reducing the CPU overhead, memory consumption, and latency.

The intuition behind the win-based method is that characteristics of DDoS attacks are observable at the traffic level (without distinguishing flow identity), making features of the whole traffic a possible substitute for flow-level features. Note that by "traffic level", we mean taking all packets together as a whole, regardless of their flow identity. In order words, the win-based method just regards all packets as a whole and samples a fixed number of packets from the whole traffic to calculate the feature values. In more detail, when DDoS attacks happen, the feature statistics of the whole traffic will be changed by the significant impact of DDoS attacks. In that case, the feature statistics of the whole traffic will present some anomalous pattern that differs from its normal state. It will be possible for machine learning to distinguish between the normal state of the whole traffic and the state when it is under a DDoS attack, by learning from the feature statistics of the whole traffic. Our proposed win-based method is designed to extract the feature statistics from samples of the whole traffic. Note that the win-based method is quite different from the flow-based method, especially for how it views the network traffic. The flow-based method looks into the details of each flow (identified by 5-tuple in the packet header, i.e., protocol type, IP source address, source port number, IP destination address, and destination port number), while the win-based method takes all the traffic as a whole to extract features without the process of distinguishing flows. Obviously, the win-based method is more efficient than the flow-based method. The overall computation overhead and detection delay of ML models will hopefully be reduced by replacing the flow-based method with the win-based method in the feature extraction phase. This replacement does not require additional changes in subsequent stages (training and validation). Thus, the use of the win-based method can also keep the compatibility with ML algorithms in the training and validation phase.

The win-based feature extraction method has the following characteristics: well-controlled overhead, wide support of commodity routers, and preserved accuracy.

• Well-controlled overhead. The overhead of extracting a set of predefined features from data is controlled by parameter $W$, with smaller $W$ leading to smaller overhead. Note that $W$ should be selected carefully, because selecting a $W$ that is not too big while containing enough information is the key to reducing overhead and preserving detection accuracy (experiments on the impact of $W$ is explained in Subsection 5.3). Once the value of $W$ is determined, the amount of packets to be processed in a fixed time window remains stable and the overhead of feature extraction is fully under control, even in extreme cases when the traffic rate changes dramatically under DDoS attacks.

• Wide support of commodity routers. The win-based method does not involve complex sampling techniques or consume lots of computing resources, making it possible to deploy the method in resource constrained network devices such as home routers and gateway routers.

• Preserved accuray. Even though the win-based method is simple to implement and cost-efficient, it can preserve the accuracy of DDoS detection, which will be explained in the results of extensive experiments in Section 5.

Besides flow-based and our proposed win-based methods, there are some other feature extraction methods in literature, such as extracting features directly from all packets captured periodically [15], or extracting features from sampled packets obtained by other sampling techniques. Extracting features directly from all packets captured periodically suffers from the overhead caused by increased traffic rate which is especially serious and overwhelming when a DDoS attack happens, similar to the case of the flow-based method that also uses all captured packets. Sampling techniques such as systematic sampling, 1-out-of-N sampling, and uniform probabilistic sampling studied in [20], can also improve performance. However, those techniques sample one packet out of every $N$ packets, which means the number of sampled packets can still be very large under a preset parameter $N$ when a huge traffic burst occurs under a DDoS attack. In our method, the number of processed packets per second (time window) is a constant $W$, while that of aforementioned sampling techniques will linearly grow with the traffic rate. When a DDoS attack is launched, the traffic rate will increase sharply, which will lead to a sharp increase in the overhead of packet processing for those sampling techniques, while the overhead of our method remains unaffected because the window size $W$ is irrelevant to the traffic rate. Thus, our proposed method is more suitable for the DDoS detection in resource-constrained devices, where servers or routers need to run the detection module while busy handling overwhelming traffic pressure caused by the attack.

We also note that some Routers on the Internet usually capture packets based on timeout and max-packets parameters, e.g., [21, 22], in which cases our win-based method is directly supported.

Given the simplicity and versatility of our proposed win-based method, the remaining question is whether detection accuracy will be preserved, which we will analyze in Sections 4 and 5.

## 4 Comparison with typical schemes

In this section, we comprehensively review multiple typical ML-based DDoS detection schemes in literature and compare our newly proposed method with them, as shown in Table 1 [7–9, 15–19, 23–31]. We select and analyze representative ones from numerous related DDoS detection schemes to illustrate key points, not aiming to cover all the studies. The comparison includes the following aspects.

• Detection mode: Online (or real-time) detection mode captures attacks immediately after packet collection, while the offline mode requires additional static analysis after finishing data collection. Off-line mode belongs to post-incident investigation or detection, which cannot detect attacks immediately while they are happening. Online detection is more powerful with the ability to detect attacks in real time, and also more challenging in system design and implementation.

• Detection location: Detection location refers to the place that ML-based models are deployed. Detection deployed in the destination server (denoted as dst) is widely adopted in the existing schemes, since the victim side can provide more attack information to achieve high detection accuracy. In contrast, detection deployed in the middle of the network or intermediate network (denoted as mid) can detect and respond to DDoS attacks in advance before the attack packets arrive at the protected server, which is a significant advantage in many scenarios. However, detection deployed in the middle of the network also incurs additional processing overhead at routers, which is the challenge that our proposed win-based method aims to address. The deployment location should be determined according to the actual situation and requirements, and the proposed win-based method is compatible with both locations due to its cost-efficiency.

• Attack types: This column indicates the attack types detected in the respective schemes. As illustrated in Table 1, each scheme detects specific attack types in its experiments. Common attack types include TCP SYN flood, Internet control message protocol (ICMP) flood, DNS amplification, user datagram protocol (UDP) flood, and bandwidth consuming attacks. Most of the attacks are semantic ones, exploiting some weakness in certain protocols, such as TCP SYN flood and DNS amplification. Some of them are brute-force ones, such as computing resource consuming and bandwidth consuming attacks.

**Table 1** Comparison of typical ML-based DDoS detection schemes

| Scheme | Detection mode | Detection | Attack type | Acc/DR (%) | ML algorithm |
|---|---|---|---|---|---|
| [16] | offline | dst | Neptune, Land, Pod, Smurf, Teardrop | 99.81 | DT, OneR[a)], BN, RF, NB |
| [7] | online | dst | ICMP flood, TCP SYN flood, UDP flood | 97 | DT |
| [18] | offline | dst | ICMP flood, TCP SYN flood, UDP flood, Smurf | 99.8 (DR) | DT, BN |
| [8] | offline | dst | DDoS attacks on Cloud computing environment | 99 | SVM, NB, RF |
| [23] | offline | dst | UDP flood, HTTP flood, Smurf, SIDDOS | 98.63 | MLP[b)], NB, RF |
| [17] | online | dst | Computing resource consuming, bandwidth consuming | 99.98 | MLP |
| [15] | online | dst | TCP SYN flood or UDP/ICMP based attacks | >96 | RBF |
| [24] | online | dst | Ping/Faked UDP/Http flood | – | MLP |
| [9] | online | dst | TCP SYN flood | – | NB, DT, RF |
| [25] | online | mid/dst | DDoS attacks in SIP ecosystems | 89–92 | SMO[c)], NB, DT, RF |
| [19] | online | – | ICMP flood, TCP SYN flood, UDP flood | 99.11 (DR) | SOM[d)] |
| Ours | online + offline | mid/dst | DNS amplification, TCP SYN flood, Bandwidth consuming | 99.82 | Bagging[e)], NB, IBK[f)], DT, MLP, BN, OneR, RBF, RF, SMO |

a) One rule (i.e., OneR) [26], a classification algorithm based on rule generation and selection.
b) Multilayer perceptron (i.e., MLP) [27], a supervised learning algorithm.
c) Sequential minimal optimization (i.e., SMO) [28], a fast algorithm for training support vector machines.
d) Self-organizing map algorithm (i.e., SOM) [29], an unsupervised machine learning technique.
e) Bootstrap aggregating (i.e., Bagging) [30], a machine learning ensemble meta-algorithm.
f) Instance-based learning with parameter K (i.e., IBK) [31], an implementation of k-nearest neighbor algorithm in Weka tool.

• Acc/DR: Results of average accuracy (or detection rate) are obtained from those typical schemes. Note that Refs. [18, 19] used DR (detection rate, which is equivalent to Recall) as the basic metric to measure the effectiveness of their classifier, while other studies used accuracy (a.k.a., Acc) as the basic metric. The experimental result of accuracy for our proposed method is also given.

• ML algorithms: This column lists the machine learning algorithms used in each respective scheme.

As is shown in Table 1, detection at the victims (i.e., dst) is more commonly used. Our proposed method is compatible with both locations. DT, RF, NB and radial basis function (RBF) [32] are popular algorithms used in different studies. The detection accuracy is very high in nearly all the studies, except in session initiation protocol (SIP) ecosystems, which indicates there still exist challenges in particular scenarios. Our newly proposed method achieves relatively high accuracy (99.82% on average) based on experiments with ten ML algorithms, details of which will be explained in Section 5.

## 5 Performance evaluation

In this section, we aim to answer the following questions by conducting extensive experiments on real DDoS attack datasets.

• How does the win-based method compare to the flow-based method when using different ML algorithms for DDoS detection?

• How does window size affect the performance of the win-based method?

• How does the win-based method compare to the flow-based method when evaluated with different attack types? Does the win-based method effectively reduce feature extraction delay in online detection?

We conduct three experiments to answer the above questions, respectively. The first experiment (i.e., Figures 4 and 5 in Subsection 5.2) is done in an offline mode, and the second (i.e., Figure 6 in Subsection 5.3) and third (i.e., Table 3 in Subsection 5.4) experiments are conducted in an online

**Table 2** Description of selected features

| Feature name | Description |
|---|---|
| pkt-rate | Number of packets per second |
| byte-rate | Number of bytes per second |
| avg-id | Mean value of the id field in IP header |
| std-id | Standard deviation of id field in IP header |
| avg-ttl | Mean value of the ttl field in IP header |
| std-ttl | Standard deviation of the ttl field in IP header |
| avg-pkt-size | Average packet size |
| std-pkt-size | Standard deviation of packet size |
| avg-src-num | Mean value of source port number (TCP or UDP) |
| std-src-num | Standard deviation of source port number (TCP or UDP) |
| avg-dst-num | Mean value of destination port number (TCP or UDP) |
| std-dst-num | Standard deviation of destination port number (TCP or UDP) |
| avg-seq-num | Mean value of sequence number (TCP) |
| std-seq-num | Standard deviation of sequence number (TCP) |
| avg-ack-number | Mean value of acknowledge number (TCP) |
| std-ack-number | Standard deviation of acknowledge number (TCP) |
| avg-win-size | Mean value of window size (TCP) |
| std-win-size | Standard deviation of window size (TCP) |

(real-time) mode by replaying the traffic data with the tcpreplay tool in a simulation network through Mininet, where we capture the packets at the entrance of the victim's subnet for attack detection.

## 5.1 Experiment setup

### 5.1.1 *Datasets and selected features*

We use real trace datasets containing both benign and attack traffic for evaluation, with each dataset containing one type of DDoS attack. The first dataset contains a TCP SYN flood attack, the second dataset contains a DNS amplification attack, and the third dataset contains an attack that consumes bandwidth resources. The selected three types of DDoS attacks are frequently launched by real-world attackers.

To provide comprehensive feature information, we select 18 features that are frequently used for DDoS detection in literature, as shown in Table 2. The full feature set includes packet rate, byte rate, and mean values and standard deviations of eight fields in the headers of packets. These fields include ID, TTL, Length, in the IP packets, source port number and destination port number in UDP or TCP packets, sequence number, acknowledge number, and window size in the TCP packets. If some fields are not contained in captured packets, they are set to default values (zeros). Obviously, features such as packet rate and byte rate are probably more useful for detecting DDoS attacks. The reason for using the mean values and standard deviations of fields in headers of network and transport layers is, intuitively, the distribution of these fields may be different between certain anomalous traffic and benign traffic. Note that even though the original feature sets used for comparing win-based and flow-based methods are identical with the same feature names, the calculated feature values will be quite different for both methods because the calculation is conducted with different scopes. The flow-based method calculates features from each flow, while the win-based method just regards all packets as a whole and samples a fixed number of packets to calculate the feature values.

To make a fair comparison between win-based and flow-based methods, we do not manually filter out insignificant features for each method by predefined human knowledge or fine-tune the parameters of ML models to their best performance. Instead, we just provide the model with the full feature set from which significant features will be selected by the feature selection algorithm (i.e., Chi-square) in an automatic way, and leverage the default parameters of the Weka tool (Waikato environment for knowledge analysis, a software with a collection of machine learning algorithms for data mining tasks) to do the training and then observe the detection performance. The Weka tool, in which all the parameters are set to default values, is used for evaluation. What we want to see is the performance of each method on average without hard fine-tuning, which reflects the basic effectiveness of each method in a relatively fair manner. As a supplement, we also give the performance comparison of the algorithms after parameter tuning.

Ten-fold cross validation is applied in each experiment. The initial dataset is divided into 10 subsets, one for testing and the rest for training. Each subset is validated once, and we measure the average results of 10 runs.

#### 5.1.2 *Metrics*

We evaluate the results by metrics including Accuracy, Precision, Recall, false positive rate (FPR), feature extraction time (FET), training latency (TL), detection latency, and memory usage.

Most of existing methods used different evaluation metrics for performance measurement, making the comparison unpractical. For a fair comparison, we use multiple frequently adopted metrics for comprehensive evaluation.

- Accuracy reflects the overall effectiveness of the classification.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}.$$

- Precision is the correct rate of the detected positive (attack) samples.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

- Recall indicates the ability of the classifier to detect positive (attack) samples out of all real positives. It is also called DR in some related studies.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

- FPR reflects the wrongly classified negative (normal) samples.

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

- FET is defined as the time cost for extracting the feature vectors.
- TL is defined as time for training an ML model.
- DL is defined as duration from the time when raw packets are collected to the time when the detection result is given.
- Memory usage measures the resource consumption of an algorithm. We use the time tool to measure the memory usage.

### 5.2 Performance comparison with different ML algorithms

In the first experiment, we compare the performance between win-based and flow-based methods using different ML algorithms. This experiment is done in an offline mode. We provide both methods with the same feature set and conduct the feature selection automatically by Chi-square algorithm with default parameters in the Weka tool. An overview of the performance comparison between the win-based and flow-based methods on six metrics is given in Figure 4 (note that J48 shown in the figure is a classification algorithm based on a decision tree).

Figure 4(a) shows that the accuracy differences between win-based and flow-based methods are less than 0.5% for most algorithms except for RBF and SMO. The default parameters set by Weka tool may not be good enough for RBF and SMO, which leads to a big accuracy gap for both algorithms. By re-adjusting parameters in further experiments, we find the accuracy gap can be narrowed close to 0.5%, showing that win-based method is comparable to flow-based method in terms of accuracy, with only a small difference. Figure 4(b) shows that the precision of the two methods is very close to each other for most ML algorithms except for NB. The precision of flow-based method with NB algorithm is really bad using default parameter set by Weka tool. Just like the parameter adjustment we have done in improving the accuracy for RBF and SMO, we find that using the kernel estimator can help the flow-based NB model obtain a comparable precision of 99.9%, which shows that both methods are almost evenly matched in precision. From Figure 4(c), we can see that the recall of win-based methods is superior to the flow-based ones for most ML algorithms. This indicates that there are more attack samples not detected in flow-based methods than win-based ones.

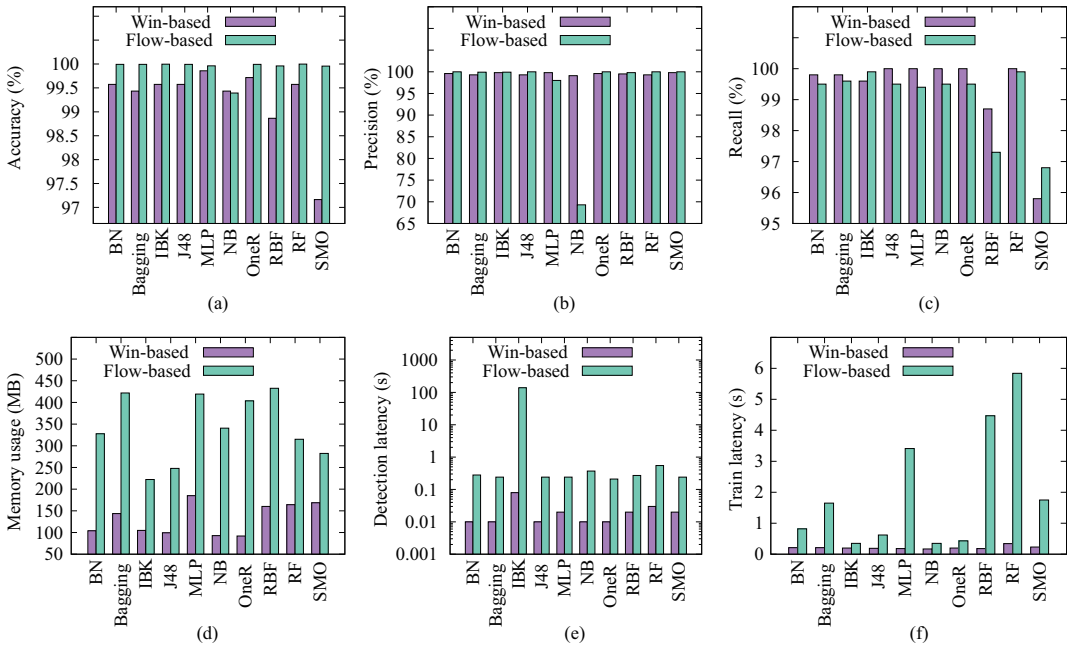


**Figure 4** (Color online) Performance comparison by using typical ML algorithms (under default parameters) with win-based and flow-based methods. (a) Accuracy; (b) Precision; (c) Recall; (d) memory usage; (e) detection latency; (f) training latency.

As shown in Figure 4(d), win-based method has a distinct advantage in memory usage with much lower values (less than half of the memory usage for flow-based method), outperforming flow-based method for all tested ML algorithms. Figure 4(e) shows that win-based method costs less than one-tenth of the detection latency (with a focus on measuring the model inference part in the first experiment) of flow-based method for all tested ML algorithms. Figure 4(f) shows that the training latency of win-based method is much lower than that of flow-based method, significantly reducing training time. Specifically, the memory usage of win-based method is kept at a very low level, only about 120 MB on average. The average detection latency is 46 ms for win-based method and 764 ms for flow-based method, with a difference of 16 times. Train latency, 0.9 s on average for the flow-based method, is 3 times that of the win-based one (e.g., 0.3 s). The experiments demonstrate the significant performance improvement achieved by win-based methods.

To comprehensively evaluate both methods, we also conduct performance comparison for win-based and flow-based methods based on fine-tuned parameters. We optimize the parameters of ten machine learning algorithms, and compared the best performance obtained by the ten algorithms under the flow-based method and the win-based method. As shown in Figure 5, parameter tuning improves the Accuracy, Precision and Recall of each ML algorithm to varying degrees, whether it is used with win-based or flow-based method. In particular, there are some significant improvements achieved by parameter tuning, such as the Accuracy of SMO with win-based method (increased from 97.2% to 99.7%), Precision of NB with flow-based method (increased from 69.3% to 99.9%), and Recall of SMO with win-based method (increased from 95.8% to 99.9%). In general, the performance of win-based and flow-based method for those ten ML algorithms is approximately on the same level and very close to each other. For accuracy, win-based method lags behind flow-based with a gap smaller than 0.6%. For Precision, the gap is smaller than 0.7%. For Recall, win-based method achieves better results than flow-based method by 0.1% to 3.2% for most of the evaluated algorithms. The experiment of performance comparison based on fine-tuned parameters further demonstrates that the performance for accuracy metrics of win-based method is comparable to that of flow-based method.

We also calculate the FPR for both methods. We note that the mean FPR of win-based method is 0.88%, which is worse than 0.06% of the flow-based one. In spite of the difference, the FPR of win-based method remains at a low level and acceptable. The gaps between the two methods in accuracy and FPR are reasonable, because the fact win-based method is much simpler and uses fewer packets to produce features than flow-based method.

By comparison with flow-based method, we find that win-based method achieves comparable perfor-

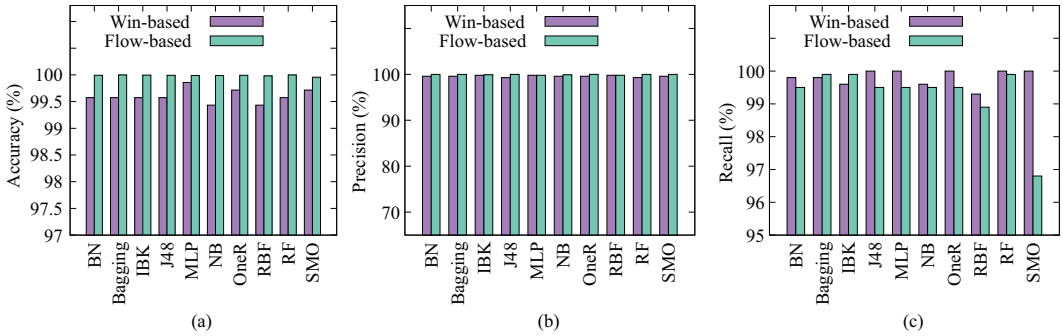


**Figure 5** (Color online) Performance comparison of ML algorithms (after tuning parameters) used with win-based and flow-based methods. (a) Accuracy; (b) Precision; (c) Recall.

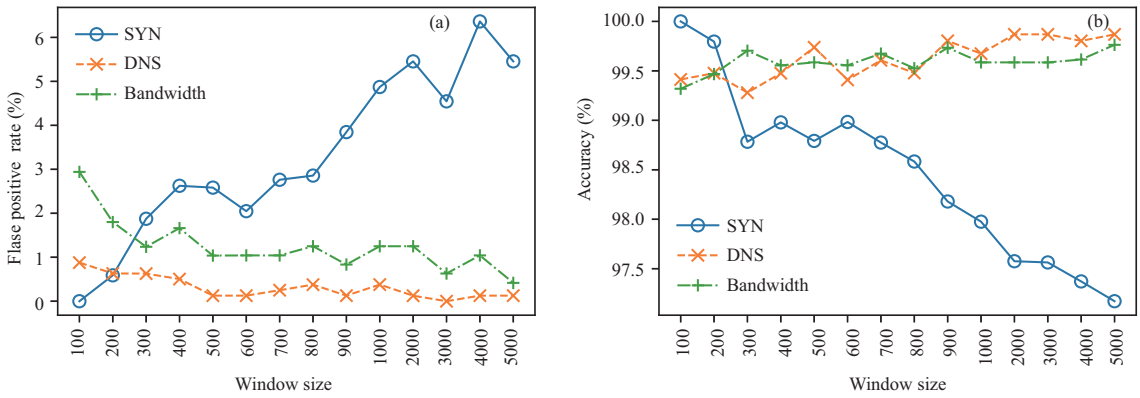


**Figure 6** (Color online) Performance tuning by varying the window size. (a) FPR; (b) Accuracy.

mance in Accuracy, FPR, Precision and Recall in most cases. Even though the win-based method is a little behind flow-based method in some metrics when applying specific algorithms, its performance still remains at good and acceptable level, showing its competence of detecting DDoS attacks. Notably, win-based method achieves significant reduction in memory usage, detection latency, and training latency, showing remarkable improvement in reducing overhead.

### 5.3 Impact of window size

The second experiment presents how window size affects the accuracy and FPR of the detection in the win-based method. The experiment is conducted in online mode.

We explore the impact of the window size with random forest algorithm, which is a commonly used classifier. The result is shown in Figure 6. As shown in Figure 6, the accuracy drops down and the FPR rises up with the increase of the window size in the experiment with the first dataset. In the experiments with the second and third datasets, the accuracy and FPR are more stable than those with the first dataset, and the accuracy and FPR tend to be better with the increase of the window size. One possible explanation for the difference is that the distributions of the attack traffic in those datasets are different. In the first dataset, the attack traffic is sparse, which results in that the performance of random forest algorithm becomes unstable when window size varies. The fluctuation of the number in the other two datasets is much smaller than the first one. On the whole, the accuracy remains at a high level and FPR remains at a low level when window size varies.

As illustrated in Figure 6, we can see that the FPR is relatively low, and the accuracy is averagely high when the window size is 200. Note that this value is also used as default window size in the first and the third experiments.

### 5.4 Performance comparison on different attack types

In the third experiment, we compare the win-based and flow-based methods on different attack types by conducting an online experiment. As is illustrated in Table 3, the first dataset is used to evaluate the performance on the attack type of TCP SYN flood, the second dataset is used to evaluate the performance

**Table 3** Comparison of two methods on different attack types with RF algorithm (online detection)

| Dataset No. | Attack type | Method | Accuracy (%) | FPR (%) | FET (s) |
|---|---|---|---|---|---|
| 1 | TCP SYN flood | win-based | 99.60 | 0.000 | 1.82 |
| | | flow-based | 99.85 | 0.303 | 45.29 |
| 2 | DNS amplification | win-based | 99.78 | 0.435 | 8.44 |
| | | flow-based | 100.00 | 0.000 | 151.25 |
| 3 | Bandwidth consuming | win-based | 99.05 | 0.483 | 19.40 |
| | | flow-based | 99.99 | 0.001 | 541.31 |

on the attack type of DNS amplification, and the third dataset is used to evaluate the performance on the attack type of bandwidth consuming.

The metrics we compare include Accuracy, FPR, FET. As detection latency is defined as the duration from the time when raw packets are collected to the time when the detection result is given, it means that detection latency consists of two parts: the time for online feature extraction time and the time for model inference (i.e., prediction or classification). Our proposed win-based method is intentionally designed to reduce the time of feature extraction rather than the time of model inference, so we focus on measuring the improvement of FET achieved by the win-based method, which is also the key factor that enables the reduction of the overall detection latency. Note that the results of FET in Table 3 are calculated as the accumulated feature extraction time for the whole detection period of each dataset.

In the win-based method, the window size is set as 200 which is a parameter that works well in the second experiment, and packets within the window are captured every 0.5 s. In the flow-based method, in order to guarantee that there are enough packets in a flow, the number of packets captured is 50000 before each feature vector is generated, which is much more than that of the win-based method.

The results in Table 3 illustrate that the accuracy and FPR with the win-based method are close to those with the flow-based method regardless of the attack types. The number of packets captured with the flow-based method is far more than that with the win-based method, which leads to the increase in the processing overhead of the flow-based method. The accumulated FET of the flow-based method is nearly 18 times to 28 times higher than that of the win-based method, which indicates that the win-based method greatly reduces FET and contributes greatly to the reduction of the overall detection latency in online detection. The FETs of the win-based method remain stable and keep at a very low level.

The experimental results demonstrate that the win-based method is able to reduce the FET while well maintaining good accuracy and low-level FPR, when compared to the traditional flow-based method. The win-based method reduces the number of packets that have to be captured and handled to an appropriate level while keeping the ability of distinguishing the pattern of DDoS traffic, so that the computation overhead and detection overhead are effectively reduced with ensured accuracy.

## 6 Discussion of future directions

Based on our observation, there are several open issues in ML-based DDoS detection, in particular, model update, identifying features, and adapting DL models for attack detection.

**Model update:** Training a model is a resource-intensive and time-consuming task, while attack detection is time-sensitive. However, the model update is obviously necessary when sophisticated, well-organized and new DDoS attacks emerge. Algorithms like J48, a type of decision tree, are not applicable to model extension. To better respond to dynamic attacks, incremental updating methods, e.g., incremental learning (IL), is necessary for DDoS detection. IL methods do not need to completely rebuild the model, with the potential of continuously extending the current model's knowledge with new data at a relatively low cost, which is promising and worth further studying in the DDoS detection scenario.

**Identifying features:** The features of ML models are usually manually selected and hard-coded based on expertise. When a new type of attack occurs, the ML algorithm needs adaption by experts and new features have to be constantly added to the original system, which results in increase of the system complexity. Operators should frequently retrain the model, otherwise detection accuracy cannot be ensured. According to our study, common ML-based methods for DDoS detection are mostly built upon manually selected features, which heavily depends on expert knowledge. It is meaningful to develop methods that are capable of discovering new features automatically from raw data instead of using several known ones for detection. Even though some representation learning techniques, or neural networks such

as CNN (convolutional neural network) can take raw data as input and learn features from data, the effectiveness of those techniques for learning meaningful features from network traffic data and detecting DDoS attacks remains unclear. Still, training these neural networks consumes lots of resources, impeding feasible deployment. Related questions still call for further research.

**Adapting DL models for attack detection:** DL models perform better than traditional machine learning algorithms in many tasks (e.g., digital recognition [33], emotion recognition [34], and friendship inference [35]), while DL also requires more computing resources to train and execute. Many network gateways and router devices, do not have enough computing resources (memory, processing power) for training and deploying DL models. How to give full play to the potential of DL in DDoS attack detection, is a valuable question worth studying. One challenge lies in how to design low-cost DL models that is resource compatible for deployment in resource constrained devices. Ref. [36] made initial efforts in designing a light-weight neural network-based online NIDS using Autoencoder, which shows the potential of this research direction. Furthermore, due to the lack of interpretability, DL models may be vulnerable in adversarial environments [33, 34, 37]. How to strengthen the robustness of DL models with objective constraints in the field of DDoS attack detection is a valuable direction.

## 7 Related work

DDoS has been one of the hot research topics for a long time [38]. A lot of DDoS variants have been reported, such as flooding (e.g., SYN, ICMP), amplification (e.g., DNS), and other DDoS attacks based on different protocols [14]. The academic community has come up with numerous DDoS defense proposals, such as traffic scrubbing which is widely adopted by industry [1)], spoof detection [39], IP traceback [40], blackholing [41], mitigation with programmable switches (P4) [42,43], and game-theoretic deterrence [44]. In addition, some researchers propose next-generation network architectures that are designed to enhance the ability to defend against DDoS, such as SCION [45] and SAVI [46]. Ref. [47] categorized the existing DDoS flooding attacks and provided a comprehensive survey of defense mechanisms.

With the popularity of machine learning, lots of efforts have been made in studying DDoS detection by machine learning, and our work falls into the ML-based DDoS detection category. Selected related studies of ML-based DDoS detection have been analyzed in Table 1 and a comprehensive comparison among them has been made in Section 4. Each of those studies makes efforts to detect one or several types of DDoS attack by applying different ML algorithms on collected traffic data, and by using typical machine learning steps such as feature extraction, model training, and testing. Ref. [8, 16, 18, 23] detected attacks in an offline mode, and Ref. [7,9,15,17,19,24,25] could be used for online DDoS detection. Ref. [19] presented a lightweight method for DDoS attack detection based on traffic flow features using NOX/OpenFlow. Ref. [9] proposed an approach for SYN flood attack mitigation based on supervised learning classification methods. Different from those studies, our work focuses on reducing feature extraction overhead by proposing a win-based feature extraction method, aiming to make real-time ML-based DDoS detection more scalable and easier for deployment on common resource-constrained network devices.

## 8 Conclusion

This article systematically studies various ML-based DDoS detection methods using flow-based features. We observe that the flow-based feature extraction method incurs remarkable detection delay and computation overhead. To better adapt ML-based DDoS detection methods to resource constrained network devices, we propose a new window-based method to reduce detection delay and computation overhead. The proposed win-based method has the advantages of well-controlled overhead, wide support of commodity routers, and preserved accuracy. We evaluate and compare the proposed win-based method with the commonly used flow-based method, by extensive experiments using ten typical ML algorithms on real datasets. The experimental results show that the win-based method effectively improves efficiency with reduced detection delay and computation overhead while ensuring detection accuracy, which can further promote the deployment of ML-based DDoS detection methods in real-world resource constrained network devices. We also analyze the impact of window size and demonstrate the effectiveness of the

1) Akamai cloud security for DDoS protection. 2021. https://www.akamai.com/us/en/products/security/ddos-protection-service.jsp.

win-based method under different attack datasets. Moreover, we discuss the challenges and research directions in this area.

**Acknowledgements** This work was supported in part by China National Funds for Distinguished Young Scientists (Grant No. 61825204), National Natural Science Foundation of China (Grant Nos. 61932016, 62132011, 62202258), Beijing Outstanding Young Scientist Program (Grant No. BJJWZYJH01201910003011), China Postdoctoral Science Foundation (Grant No. 2021M701894), China National Postdoctoral Program for Innovative Talents, and Shuimu Tsinghua Scholar Program.

**References**

1 Antonakakis M, April T, Bailey M, et al. Understanding the mirai botnet. In: Proceedings of the 26th USENIX Security Symposium (USENIX Security), 2017. 1093–1110
2 Zheng J, Li Q, Gu G, et al. Realtime DDoS defense using COTS SDN switches via adaptive correlation analysis. IEEE Trans Inform Forensic Secur, 2018, 13: 1838–1853
3 Wang C, Miu T T N, Luo X, et al. SkyShield: a sketch-based defense system against application layer DDoS attacks. IEEE Trans Inform Forensic Secur, 2017, 13: 559–573
4 Xiao B, Chen W, He Y, et al. An active detecting method against SYN flooding attack. In: Proceedings of the 11th IEEE International Conference on Parallel and Distributed Systems (ICPADS), 2005. 709–715
5 Kambourakis G, Moschos T, Geneiatakis D, et al. Detecting DNS amplification attacks. In: Proceeding of International Workshop on Critical Information Infrastructures Security, Berlin, 2007. 185–196
6 Sun C, Liu B, Shi L. Efficient and low-cost hardware defense against DNS amplification attacks. In: Proceedings of IEEE Global Telecommunications Conference (GLOBECOM), 2008. 1–5
7 Chen Y W, Sheu J P, Kuo Y C, et al. Design and implementation of IoT DDoS attacks detection system based on machine learning. In: Proceedings of IEEE European Conference on Networks and Communications (EuCNC), 2020. 122–127
8 Wani A R, Rana Q P, Saxena U, et al. Analysis and detection of DDoS attacks on cloud computing environment using machine learning techniques. In: Proceedings of Amity International Conference on Artificial Intelligence (AICAI), 2019. 870–875
9 Degirmencioglu A, Erdogan H T, Mizani M A, et al. A classification approach for adaptive mitigation of SYN flood attacks: preventing performance loss due to SYN flood attacks. In: Proceedings of IEEE/IFIP Network Operations and Management Symposium, 2016. 1109–1112
10 Radware. DDoS protection & DDoS mitigation solutions. 2021. https://www.radware.com/solutions/ddos-protection/
11 Safavian S R, Landgrebe D. A survey of decision tree classifier methodology. IEEE Trans Syst Man Cybern, 1991, 21: 660–674
12 Biau G. Analysis of a random forests model. J Mach Learning Res, 2012, 13: 1063–1095
13 Rish I. An empirical study of the naive Bayes classifier. In: Proceedings of IJCAI Workshop on Empirical Methods in Artificial Intelligence, 2001. 41–46
14 Mirkovic J, Reiher P. A taxonomy of DDoS attack and DDoS defense mechanisms. SIGCOMM Comput Commun Rev, 2004, 34: 39–53
15 Karimazad R, Faraahi A. An anomaly-based method for DDoS attacks detection using RBF neural networks. In: Proceedings of the International Conference on Network and Electronics Engineering (ICNEE), 2011. 44–48
16 Osanaiye O, Choo K K, Dlodlo M. Analysing feature selection and classification techniques for DDoS detection in cloud. In: Proceedings of Southern Africa Telecommunication, 2016. 198–203
17 Barati M, Abdullah A, Udzir N I, et al. Distributed denial of service detection using hybrid machine learning technique. In: Proceedings of International Symposium on Biometrics and Security Technologies (ISBAST), 2014. 268–273
18 Wang W, Gombault S. Efficient detection of DDoS attacks with important attributes. In: Proceedings of International Conference on Risks and Security of Internet and Systems (CRiSIS), 2008. 61–67
19 Braga R, Mota E, Passito A. Lightweight DDoS flooding attack detection using NOX/OpenFlow. In: Proceedings of IEEE Local Computer Network Conference (LCN), 2010. 408–415
20 Korczynski M, Janowski L, Duda A. An accurate sampling scheme for detecting SYN flooding attacks and portscans. In: Proceedings of IEEE International Conference on Communications (ICC), 2011. 1–5
21 Huawei. HUAWEI: configuring the device to capture packets. 2021. https://support.huawei.com/enterprise/en/doc/EDOC1100112354/bd0e10ad/configuring-the-device-to-capture-packets
22 Cisco. Cisco: configure packet capture on Cisco SCE 8000. 2021. https://www.cisco.com/c/en/us/support/docs/service-exchange/service-control-operating-system-software/200464-Packet-caputre-on-Cisco-SCE-8000.html
23 Alkasassbeh M, Al-Naymat G, Hassanat A, et al. Detecting distributed denial of service attacks using data mining techniques. Int J Adv Comput Sci Appl, 2016, 7: 436–445
24 Seufert S, O'Brien D. Machine learning for automatic defence against distributed denial of service attacks. In: Proceedings of IEEE International Conference on Communications (ICC), 2007. 1217–1222
25 Tsiatsikas Z, Geneiatakis D, Kambourakis G, et al. Realtime DDoS detection in sip ecosystems: machine learning tools of the trade. In: Proceedings of International Conference on Network and System Security (NSS), 2016. 126–139
26 Holte R C. Very simple classification rules perform well on most commonly used datasets. Machine Learn, 1993, 11: 63–90
27 Ruck D W, Rogers S K, Kabrisky M, et al. The multilayer perceptron as an approximation to a Bayes optimal discriminant function. IEEE Trans Neural Netw, 1990, 1: 296–298
28 Zeng Z Q, Yu H B, Xu H R, et al. Fast training support vector machines using parallel sequential minimal optimization. In: Proceedings of IEEE International Conference on Intelligent System and Knowledge Engineering (ISKE), Xiamen, 2008. 997–1001
29 Vesanto J, Alhoniemi E. Clustering of the self-organizing map. IEEE Trans Neural Netw, 2000, 11: 586–600
30 Breiman L. Bagging predictors. Machine Learn, 1996, 24: 123–140
31 Reynolds K, Kontostathis A, Edwards L. Using machine learning to detect cyberbullying. In: Proceedings of IEEE International Conference on Machine Learning and Applications and Workshops, 2011. 241–244
32 Musavi M T, Ahmed W, Chan K H, et al. On the training of radial basis function classifiers. Neural Networks, 1992, 5: 595–603
33 Zhao Y, Xu K, Wang H Y, et al. Stability-based analysis and defense against backdoor attacks on edge computing services. IEEE Network, 2021, 35: 163–169
34 Zhao Y, Xu K, Wang H Y, et al. MEC-enabled hierarchical emotion recognition and perturbation-aware defense in smart cities. IEEE Internet Things J, 2021, 8: 16933–16945

35 Zhao Y, Qiao M N, Wang H Y, et al. TDFI: two-stage deep learning framework for friendship inference via multi-source information. In: Proceedings of IEEE INFOCOM, 2019. 1981–1989
36 Mirsky Y, Doitshman T, Elovici Y, et al. Kitsune: an ensemble of autoencoders for online network intrusion detection. In: Proceedings of Network and Distributed System Security Symposium (NDSS), San Diego, 2018. 1–15
37 Zhao Y, Xu K, Li Q, et al. Intelligent networking in adversarial environment: challenges and opportunities. Sci China Inf Sci, 2022, 65: 170301
38 Osterweil E, Stavrou A, Zhang L. 21 years of distributed denial-of service: current state of affairs. Computer, 2020, 53: 88–92
39 Jin C, Wang H, Shin K G. Hop-count filtering: an effective defense against spoofed DDoS traffic. In: Proceedings of ACM Conference on Computer and Communications Security (CCS), 2003. 30–41
40 Song D X, Perrig A. Advanced and authenticated marking schemes for IP traceback. In: Proceedings IEEE International Conference on Computer Communications (INFOCOM), 2001. 878–886
41 Dietzel C, Wichtlhuber M, Smaragdakis G, et al. Stellar: network attack mitigation using advanced blackholing. In: Proceedings of International Conference on emerging Networking Experiments and Technologies (CoNEXT), 2018. 152–164
42 Zhang M, Li G, Wang S, et al. Poseidon: mitigating volumetric DDoS attacks with programmable switches. In: Proceedings of Network and Distributed System Security Symposium (NDSS), San Diego, 2020. 1–18
43 Liu Z, Namkung H, Nikolaidis G, et al. Jaqen: a high-performance switch-native approach for detecting and mitigating volumetric DDoS attacks with programmable switches. In: Proceedings of USENIX Security Symposium (USENIX Security), 2021. 3829–3846
44 Li Y, Li H, Lv Z, et al. Deterrence of intelligent DDoS via multi-hop traffic divergence. In: Proceedings of ACM SIGSAC Conference on Computer and Communications Security (CCS), 2021. 923–939
45 Zhang X, Hsiao H C, Hasker G, et al. SCION: scalability, control, and isolation on next-generation networks. In: Proceedings of IEEE Symposium on Security and Privacy (S&P), 2011. 212–227
46 Wu J, Bi J, Bagnulo M, et al. Source address validation improvement (SAVI) framework. RFC7039. 2013. https://datatracker.ietf.org/doc/html/rfc7039
47 Zargar S T, Joshi J, Tipper D. A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. IEEE Commun Surv Tutorials, 2013, 15: 2046–2069