Engineering Science and Technology, an International Journal 35 (2022) 101176

HOSTED BY

Contents lists available at ScienceDirect

Engineering Science and Technology, an International Journal

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

Detecting flooding DDoS attacks in software defined networks using supervised learning techniques

Song Wang [a,*], Juan Fernando Balarezo [a], Karina Gomez Chavez [a], Akram Al-Hourani [a], Sithamparanathan Kandeepan [a], Muhammad Rizwan Asghar [b], Giovanni Russello [b]

[a] RMIT University, Melbourne, VIC 3000, Australia
[b] Cyber Security Foundry, The University of Auckland, Auckland 1010, New Zealand

ARTICLE INFO

Article history:
Received 28 July 2021
Revised 15 April 2022
Accepted 11 May 2022
Available online 3 June 2022

Keywords:
Machine learning
Software Defined Networks (SDN)
Distributed Denial of Service (DDoS)
OpenFlow
Network Security

ABSTRACT

For the easy and flexible management of large scale networks, Software-Defined Networking (SDN) is a strong candidate technology that offers centralisation and programmable interfaces for making complex decisions in a dynamic and seamless manner. On the one hand, there are opportunities for individuals and businesses to build and improve services and applications based on their requirements in the SDN. On the other hand, SDN poses a new array of privacy and security threats, such as Distributed Denial of Service (DDoS) attacks. For detecting and mitigating potential threats, Machine Learning (ML) is an effective approach that has a quick response to anomalies. In this article, we analyse and compare the performance, using different ML techniques, to detect DDoS attacks in SDN, where both experimental datasets and self-generated traffic data are evaluated. Moreover, we propose a simple supervised learning (SL) model to detect flooding DDoS attacks against the SDN controller via the fluctuation of flows. By dividing a test round into multiple pieces, the statistics within each time slot reflects the variation of network behaviours. And this "trend" can be recruited as samples to train a predictor to understand the network status, as well as to detect DDoS attacks. We verify the outcome through simulations and measurements over a real testbed. Our main goal is to find a lightweight SL model to detect DDoS attacks with data and features that can be easily obtained. Our results show that SL is able to detect DDoS attacks with a single feature. The performance of the analysed SL algorithms is influenced by the size of training set and parameters used. The accuracy of prediction using the same SL model could be entirely different depending on the training set.

1. Introduction

Software Defined Networking (SDN) has received a great attention due to its novel architecture, which provides a potential solution to the management of fast growing networks. Decoupled control and data plane, centralised controller and programmability maximise the flexibility and adaptability on the administrator side. As a three-layer model, the application layer is on the top preparing all the rules and policies defined by the administrator, these rules can be updated dynamically through an SDN controller. A modification in the application layer could change the behaviour of the entire network. Thanks to the open-source platform, the development of the application layer no longer relies on vendors alone [1]. Getting rid of the licence constraints, SDN allows administrators to develop their own applications to customise the network management over general-purpose hardware. The control layer, however, is the brain of the model, which converts rules from the application layer to readable messages towards the underlying data layer and collects feedback from the data layer backwards to the application layer. After a decision is made on the control layer, the implementation is realised by the data layer. The data layer in the SDN has no intelligence, it just follows instructions from the control plane [2]. Centralised control allows network operators to automatically manage security strategies for large scale networks. SDN simplifies the building, deployment, and maintenance of a network. The access to the network is permitted without exposing the details of the underlying layer. It is easy to enhance network features by updating SDN applications. Thus, simple hardware devices are sufficient in the network. Besides being cost-effective, SDN-based devices have almost no issues related to compatibility [3]. However, SDN also poses new security threats due to its innovative architecture, such as the cen-

* Corresponding author.
E-mail address: s3478896@student.rmit.edu.au (S. Wang).

https://doi.org/10.1016/j.jestch.2022.101176
2215-0986/

tral controller, it is a double edged sword; although it is easier to manage a network with one controller, this can lead to a single point of failure, especially under Distributed Denial of Service (DDoS) attacks [4]. The brand-new framework of SDN means existing mechanisms against cyber threats in the traditional network need to be updated in order to protect this modern platform. Consequently, SDN has not been widely proven to be a trustworthy security solution as the next generation network [5,6].

DDoS attack is a typical threat, where attackers generally compromise multiple devices that act as bots, and synchronise fake traffic or requests to deplete the system's resources or congest the victim network. Unlike other types of network attacks, a victim does not lose any sensitive information. Instead, the impact is the out-of-service experience during the attack period. DDoS attacks are relatively easy to carry out but hard to trace, *e.g.,* due to IP spoofing [7]. In general, DDoS attacks can be categorised into three types: (i) application layer attack, (ii) protocol attack, and (iii) volumetric attack. For launching an application layer attack, attackers concentrate on consuming the bandwidth and computing resources of a server, so that it is unable to support services. To launch a protocol attack, attackers drain the available state of a target by exploiting the procedure of a protocol, *e.g.,* three-way handshake process of TCP. To trigger a volumetric attack, a large amount of malicious traffic is flooded towards the target to drain the available resources, such as bandwidth and CPU utilisation [8]. DDoS attacks in the SDN data plane are similar to the traditional network, the processing capability of a switch or the links between switches could be impacted. However, when the application plane or control plane of SDN is under DDoS attack, the entire network is in trouble, because these two planes define and implement the policies in the network. In other words, they are high-profit targets for intruders to interrupt the network. Additionally, the northbound and southbound interfaces between these three planes are also vulnerable, because the interactions between SDN planes are sent through networks rather than within the same switch/router in the legacy network. In this article, we focus on the volumetric DDoS attack only, because it is the most popular DDoS attacks [9], and will be a long-term threat due to the rapid growth of connected network devices.

Since the behaviour in the SDN is a new research area, using Machine Learning (ML) to study the behaviour could be an attempt to early detection [10], as the response from a machine is usually faster than human beings. ML is a technique that allows a machine to make decisions through training, features of data, or trial and error methods. It is an auxiliary approach to help network operators realise the current situation. ML has been deployed in a wide area from medical analysis to data mining. Existing solutions usually require multiple features of the traffic, it is hard to collect this kind of dataset in the real network due to accessability or authorisation. And most of the models are verified by experimental datasets or simulations only, real testbed validation is rare. In this article, we employ supervised learning (SL) to detect DDoS attacks in the SDN. Because SL enables learning from past experiences through training, it could be more specific in the DDoS attack detection. We verify and compare the performance of different SL algorithms without multiple features picked from the training set. In theory, a single threshold can detect flooding DDoS attacks, however, it's hard to distinguish legitimate burst traffic from attack traffic. To improve the accuracy, one way is to define a DDoS attack after the threshold being hit for several consecutive times, and this allows burst traffic but delays the detection. If we can study/analyse the network traffic behaviour before setting the flooding DDoS attack definition, we might get a better accuracy and early detection. Thus, using a single feature is worth a try, and we decide to employ only the trend of the number of packets/bytes as a reference for flooding DDoS attack detection. This kind of dataset is easier to obtain and consumes less time and resources in the training phase than other models which require multiple features. And since DDoS attack detection can be achieved via a basic feature, it alleviates the issue of supervised training that high quality training set is hard to gather.

The contributions of this article are:

- It designs and implements an SDN application for DDoS attack detection based on SL.
- It tests on the real and emulated SDN network with a range of SL techniques and various traffic models.
- It evaluates and analyses the proposed model using 1999 DARPA and InSDN dataset.
- The performance, especially accuracy, of each SL technique is analysed and compared to present an overview of the output. Moreover, initialisation time and prediction of SL classifier is studied to show the overhead of deploying SL.

Test results demonstrate that (i) data preparation, SL model training and real-time prediction can all be implemented over a single SDN controller; (ii) a SL model is able to detect flooding DDoS attacks with a single feature; (iii) the ratio of anomalous sample to normal sample in the training set has limited impact on the prediction; (iv) more data and features in the training usually result in a better performance, and yet overfitting might happen when redundant features confuse the SL algorithm. Finally, the implementation over a real testbed shows that a real-time SL detection over physical SDN network is feasible.

The rest of this article is organised as follows. Section 2 reviews related works. Section 3 describes the proposed system model and the generation of training sets. Test results using experimental dataset and self-bootstrapped data are explained and compared in Sections 4 and 5, respectively. Discussion and future direction are presented in Section 6. Finally, Section 7 concludes this article.

2. Related work

The main metrics used for evaluating a ML algorithm can be computed as True Positive (T_P), True Negative (T_N), False Positive (F_P) and False Negative (F_N). In our test, positive refers to normal states and negative refers to attack states. These metrics are used to compare the performance of each ML technique computing the following indicators [25]:

- *Accuracy (A)* is the ratio of a correct prediction made by a ML algorithm, including both successful positive and negative decision.
- *Precision (P)* is defined as the ratio of a correct decision made for normal network state among all the normal state predictions.
- *Recall (R)* is the ratio of a correct decision made for normal network state among all the normal state situations.
- *F-Measure (F)* measures the quality of both precision and recall.

SL is a typical ML type that has been deployed for DDoS attack detection in several previous works, it trains a predictor with labelled data and then employs this predictor to make decisions. The performance of a SL algorithm can be validated through known test samples, so that network administrators gain confidence from validation results before deploying the predictor in the network. When applied in the classification, a SL predictor calculates the probability of each class, and choose the one with highest probability. This section analyses the literature of SL applications, including DDoS attack detection, and identifies the gaps. A summary of existing works and our proposed model is presented in Table 1 and explained in the following.

Table 1
Existing algorithms to protect SDN against DDoS attacks using SL.

Solution	SL Techniques	Experiments	Dataset	Features	Remarks
Meti et al. [11]	NB, SVM, and NN	Mininet simulation	Real-time dataset from TCP traffic	Number of hosts connected per second	SVM can be deployed as an intrusion detection system (IDS) in the SDN
Zekri et al. [12]	DT	Virtual environment on VMs	Self-generated traffic using python scripts and hping3	Protocol and service type, flag, TTL, and source/destination IP	DT has over 98% accuracy in DDoS attack detection and 0.6 s detection time
Tuan et al. [13]	K-Nearest Neighbour (KNN), DT and NN	Replay a public dataset along with self-generated traffic over a real testbed	CAIDA 2007 and self-generated traffic	Number of ports per IP, the entropy of ports per IP and number of ICMP packets per IP	A lightweight countermeasure with an adaptive monitor window time, the accuracy can be over 98%
Sahoo et al. [14]	SVM, KNN and Random Forest	Train the model with experimental dataset and run the model in Mininet	NSL-KDD and dataset in [15]	Extracted features from experimental datasets, details unmentioned	Improve the SVM model with feature extraction and parameter optimisation. The accuracy is almost 99%
Bakker et al. [16]	DA, SVM, KNN, NB, and DT	Deploy a dedicated ML device (DPAE) in the data plane to classify traffic	ISCX data	Number of bytes sent by source and destination, number of packets sent by source, flow duration, number of bytes divided by number of packets sent by source and destination	SL can be deployed in the SDN as a classifier; however, it is hard to pick features for better detection
Polat et al. [17]	SVM, KNN, NN and NB	Virtual environment on VMs	Self-generated traffic using hping3	12 features including the number of packets received on the control plane	Using feature selection to filter key features before training can improve the detection accuracy
Huyu et al. [18]	Apache Spark	Try dataset over Apache Spark system	DDoS_DNS_AMPL, DDoS_CHARGEN and RADB_DDoS	Source/destination IP, source/destination port number, protocol, packet length, number of bytes, and timestamp	Deploy monitoring system outside of the monitored network can achieve data analysis on a large number of machines
Ahmed et al. [19]	DPMM	Running DPMM to classify DDoS attacks from dataset	Dataset in [20]	Number of packets transmitted, ratio of source and destination bytes, and connection duration time	DDoS attack detection accuracy is almost 100% when the number of connections is less than 300
Dong et al. [21]	KNN, NB and SVM	Mininet simulation	Self-generated traffic using hping3	Flow length, flow duration, flow size and flow rate	Introduce a weight value in the neighbours to improve the original KNN model. The performance is remarkable from simulation.
Mohammed et al. [22]	NB	4 ISPs with 4 different SDN controllers in the physical testbed	NSL-KDD	25 features in total including protocol and duration	DDoS mitigation using the proposed model has encouraging results among ISPs
Niyaz et al. [23]	Soft-max, NN, and Stacked Autoencoder	Verified in the physical home wireless network	Self-generated traffic using tcpdump	34 features from TCP flows, 20 features from UDP flows, and 14 features from ICMP flows	Individual DDoS attack detection accuracy can be over 95%, while for normal and attack classification it reaches over 99% accuracy
Wang et al. [24]	SVM	Try dataset over physical testbed	KDD 1999, KDD CUP 1999	30 features including protocol and flag	Accuracy varies with the number of features chose to classify traffic
Proposed model	SVM, GLM, NB, DA, FNN, DT, KNN and BT	Mininet and real testbed	DARPA, InSDN and self-generated traffic	1 feature: the fluctuation of number of Packet_In messages within a period	Multiple techniques have over 99% accuracy in the simulation, and over 90% in the real time detection.

2.1. Performance of SL in DDoS attack detection

Meti et al. [11] prepare a dataset which consists of only TCP traffic from the real network, and adopt two features, which are the number of connected devices in second and peak/off-peak-time indicators, to train the SL classifiers, which are naive bayes (NB), support vector machine (SVM) and neural network (NN), with two labels: normal and abnormal. Results of comparison in accuracy, precision, and recall show that NN has the best accuracy and precision of up to 80% and 100%, respectively, while the SVM has a recall value of 80%, higher than the other two algorithms. Similarly, Zekri et al. [12] propose to use a decision tree (DT) algorithm to detect DDoS attacks in the cloud network. The splitting criterion of DT chooses the attributes with the largest gain ratio, and then the dataset is divided into subsets based on the attribute. This procedure repeats until no further branch can be generated, then a DT learnt from the training set is ready to classify incoming traffic. They define the traffic into four categories, and verify the proposed model by self-generated traffic in the simulation. Form the test, the correct classification rate is over 98% and it only takes 0.58s to make a decision. Tuan et al. [13] calculate the entropy and logarithm value to detect TCP-SYN flood and ICMP flood attacks in the SDN, respectively. KNN is recruited to find K nearest Euclidean distance points from the current entropy or logarithm point to determine whether the network is under DDoS attacks. This model is evaluated with CAIDA 2007 dataset and self-generated traffic using Bonesi, the accuracy is over 99% when K = 9. Sahoo et al. [14] propose an improved SVM model which employs kernel principal component analysis (KPCA) and genetic algorithm (GA) to detect DDoS attacks. This model periodically retrieves flow statistics from switches via the SDN controller, and then uses KPCA to extract essential features and GA to adjust parameters of SVM so as to achieve an optimal prediction. Through tests over public datasets and simulations, the accuracy of this model is almost 99%.

As early detection is becoming one of the essential features of a mitigation technique, the duration in the training and prediction of SL classifier is studied. Bakker et al. [16] compare the initialisation time and accuracy of seven classifiers to see the overhead of deploying ML for DDoS attack detection in the SDN. From an experimental dataset, some features that mainly concerned with the amount of data sent in one way and the duration of a connection are selected to train and validate the classifiers. And a 30-fold cross validation is performed to measure the performance of each classifier. The results show that SVM needs more time for initialisation

than other algorithms; however, its processing time in the detection is the shortest and the accuracy is over 93%, which is better than others. Polat et al. [17] also compare the performance of four ML techniques in the DDoS attack detection with and without feature selection. To obtain the most related features among 12 features for detection, filter-based, wrapper-based and embedded based feature selection methods are implemented on the self-generated dataset. After feature selection, the number of selected key features varies from 6 to 10 in each ML algorithm, and the accuracy of detection improves more or less when the model is trained by these key features rather than trained by all the 12 features. The best performed algorithm is KNN using wrapper-based selection, the accuracy is 98.3% when trained by 6 key features, and the original accuracy is 95.67% when trained by 12 features.

2.2. ML deployment for DDoS attack detection

During network monitoring, Huyu et al. [18] propose to pass real-time traffic data to an off-line learning pipeline for model building and optimisation. Data is collected over routers and sent to the pipeline for data transformation and feature engineering. Through this off-line style, the model is studying the intrinsic patterns from the variation of traffic in real time, and only the improved version, i.e., better accuracy, that is validated can be applied to the model update. Thanks to the flexibility of the SDN, it can be combined with an existing model to protect the network against DDoS attacks. Ahmed et al. [19] introduce an SDN-based Dirichlet Process Mixture Model (DPMM) clustering approach for DNS query-based DDoS attack detection. The two main units in this model are the traffic statistic manager and the learner component. Traffic statistic manager periodically collects traffic features, such as the duration of a connection, from the data plane through SDN controller, and these features are sent to the learner component to analyse and detect suspicious traffic. Traffic classification is almost 100%, however, the overall accuracy, which includes normal traffic classification, is under 70%. This means there will be many incorrect predictions. Dong et al. [21] assume a flow as a vector that consists of the value of flow length, duration, size and rate. The distance between two flows are defined as the square root of the difference between each feature of the flow. To improve the KNN model, a weight value is introduced to reflect the importance of a neighbour flow, a closer neighbour has a higher weight value so that it has more impact on the prediction.

The location of the ML component could be in a remote network rather than an application over the controller. Mohammed et al. [22] set up an environment with four ISPs interconnected via the Internet. They consider that a ML server based on NB classification is running outside of this network. A new incoming packet will be forwarded to the controller for feature extraction. Then, it is sent to the ML server for analysis. When the controller gets the prediction result as abnormal from the ML server, it will inform the controller in the source network to block those requests, so that the attack is stopped at the ingress of the network. Initially, the ML server learns from 25 features of traffic. This number is reduced to 18 due to the poor processing performance over real traffic. However, the precision in the normal state is less than 70%, and 92% in the abnormal state, which is not optimistic. Niyaz et al. [23] propose to run three modules, which are Traffic Collector and Flow Installer (TCFI), Feature Extractor (FE) and Traffic Classifier (TC) over the controller to analyse the traffic in the network. Apart from the responses to the new flow requests, TCFI also records packet headers for the FE. The FE computes the mean and entropy values from the features of packets and invokes the TC to classify traffic in one of the eight types, including seven kinds of DDoS attacks. From the test result, the accuracy of the model is over 95% for individual DDoS attack class, while this value reaches over 99% for a two class scenario with only normal and attack states. However, as the model has to process every packet for its feature, the system computational consumption on the controller is extremely high even in the normal state. Wang et al. [24] involve SVM in the threat detection of SDN with the help of sFlow toolset. The model collects traffic statistics from switches via the controller and extracts behavioural features, such as protocol type, for SVM training. Signature analysis is performed to learn the behavioural features of known attacks. By comparing the normal behavioural profile with the malicious ones, the attack can be detected through SVM, and test results show that the average accuracy is over 97%.

3. DDoS attack detection using SL techniques

Note that the development of new SL algorithms is beyond the scope of this article. We focus on the training and implementation of existing SL techniques [26] to detect DDoS attacks in SDN. We have used dynamic threshold in the detection in our previous works [27,28], here we implement SL to predict through the fluctuation of flows. Specifically, the investigated DDoS attack targets the SDN controller.

3.1. System architecture

Fig. 1.a and .b show the network and system architecture for offline training and DDoS attack detection using SL techniques in SDN, respectively. The network architecture is composed of SDN-switches that perform data plane operations and SDN-controller that performs control plane operations. On the top of SDN-controller, the following SDN applications are running:

- **Training and Monitoring Application (TMA).** It obtains real-time traffic statistics via the controller and convert those to the format required for ML Application in the detection phase.

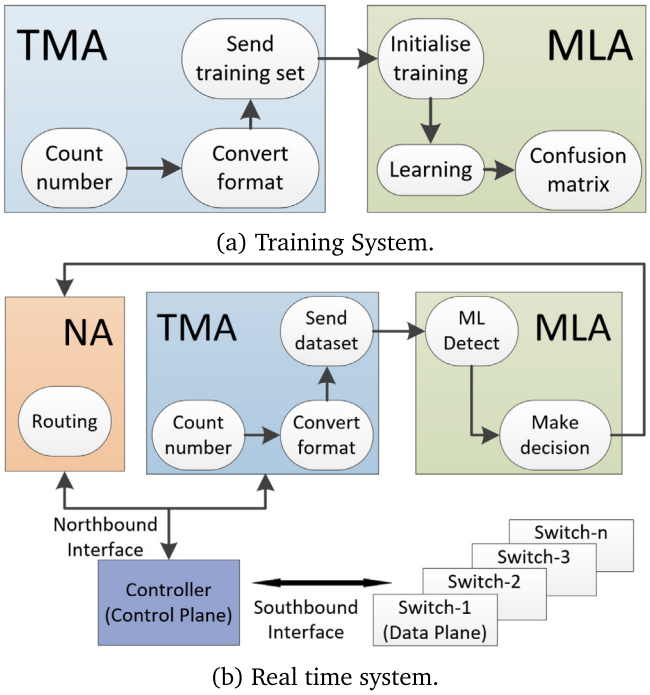


Fig. 1. The SDN architecture for DDoS training and real time detection using trained ML application.

- **ML Application (MLA).** MLA is a Matlab-based implementation. It receives the data from TMA for training and detection purposes. It is also responsible for DDoS attack detection using specific SL algorithms. MLA is integrated in the SDN application via Matlab engine API for python, it is launched during network initialisation to prepare the detector, and makes decisions in the real-time detection.
- **Network Application (NA).** NA is a normal routing application that is used to generate flow rules in the SDN switches. It is able to update flow entries according to the decision from MLA in real time. NA also receives the output of MLA and triggers required operations, for example DDoS attack mitigation.

3.2. Background of the utilised SL techniques

Among various SL algorithms, we choose eight common techniques which are under different SL categories in our test [29], they include: discriminant analysis, DT, generalised linear model, K-nearest neighbour, NB, feedforward neural network, SVM and bagging tree.

1. *Discriminant Analysis (DA):* categorises data into two or more known groups according to the combination of features [30]. Besides, DA is also used in feature extraction [31].
2. *Decision Tree (DT):* is a tree-like model whose decision rule is similar to the if-else structure in the programming language. From training, the probability of each event is studied, so that data are classified through the tree depending on its probability on each node [32].
3. *Generalised Linear Model (GLM):* is used to cover situations that have arbitrary distribution, which means the response from the model may not always be a normal distribution. Logistic regression is a typical GLM that distributes data into two discrete groups via sigmoid function [33].
4. *K-Nearest Neighbour (KNN):* is one of the simplest ML algorithms, it classifies a sample based on the number of neighbour-samples nearby. There are various distance metrics, such as Euclidean (for continuous variables) and Hamming (for discrete variables) [34].
5. *Naive Bayes (NB):* is a probabilistic classifier that makes decisions based on the overall probability given by each feature [35]. For continuous values in the DDoS attack detection, Gaussian distribution is assumed in its classification.
6. *Feedforward Neural Network (FNN):* is a simple type of neural network, it processes data in the way that is inspired by the biological neural system, decision is made through the analysis in the hidden layers of neural network [36].
7. *Support Vector Machine (SVM):* aims at finding one or multiple best hyperplanes to separate all the samples from one class to other classes. Since most of the classifications cannot be achieved via a linear solution, SVM is also able to generate a non-linear classifier with kernel function [37].
8. *Bagging Tree (BT):* uses bagging algorithm to train several weak trees from the original training set, and makes decision from the output of these trees [38]. The difference between BT and DT is that BT averages multiple DTs and reduces the variance of unstable procedures in the DT.

3.3. Working procedure

The proposed architecture is validated in the simulation and real testbed using the network topology depicted in Fig. 2. There are four aggregator hosts and one server connected to the three SDN switches. NA, TMA and MLA are running over a Ryu controller that manages and monitors the whole network. For simulation, we use Mininet emulator on the standard laptop running Ubuntu 16.04 OS with Intel i5 CPU and 8G RAM. For real implementation, the switch is Zodiac-FX switch [28], which supports OpenFlow [39] protocol, and the aggregator/server is Raspberry Pi running Debian OS, which is a linux based system. The aggregator generates periodical and random TCP/UDP traffic towards the server using a test tool, called iperf, to emulate the behaviour in the SDN. The four traffic categories are explained below:

- **Periodical UDP Traffic:** Aggregator transmits 100 bytes UDP packets every 10 s. This traffic model is usually employed as heartbeat communications between the server and the host to ensure that the host is still alive.
- **Periodical TCP Traffic.** Aggregator transmits TCP traffic using the maximum available bandwidth every 60 s. This traffic model is usually employed when a host needs to retrieve data from the far end regularly, such as reading from a healthcare device.
- **Random UDP Traffic.** Aggregator transmits 100 bytes UDP packets randomly in a window between 60–120 s. This traffic model usually happens on the DNS inquiry when a host accesses a website.
- **Random TCP Traffic.** Aggregator transmits TCP traffic using the maximum available bandwidth with a random transmission interval,this interval follows normal distribution with a mean value of 90 s and standard deviation of 30 s. This traffic model usually happens when a host downloads files from the server.

We would like to simulate the scenario that the traffic is generated in an arbitrary way, periodical and event triggered messages appear from time to time, like in the Internet of Things (IoT) network [40]. Under normal scenario, each aggregator sends one type of the above traffic to the server. Under attack scenario, aggregator-1 and aggregator-3 send periodical (TCP and UDP) and random (TCP and UDP) traffic to the server, respectively. And the other two aggregators act as compromised devices, they rapidly generate fake UDP packets whose destination does not exist using iperf, so that the requests trigger a large number of Packet_In messages towards the controller within a short period. The Packet_In message is defined in OpenFlow protocol for the interaction between the data and control plane, it is typically generated by mismatched packets. Note that the attack scenario contains both normal and malicious traffic rather than attack traffic only. Once the controller is flooded by this kind of messages, it is unable to process legitimate requests. This results in additional delay in the response to the switch, or even there is no response due to the saturation of bandwidth, as well as the shortage of computing resources. This could happen when multiple devices under the same aggregator are infected, and all of them attempt to exhaust network resources.

In the training phase, TMA collects the counter of Packet_In messages through the controller in the normal and attack scenarios to prepare the training set. The controller keeps counting the number of Packet_In messages per host every fixed duration (we call this duration a time slot), and the counter is stored in a list. The list has a new record after each time slot, and the controller will print out the list once the list has enough records. Here the number of records to trigger the print operation is predefined by the user, it is 30 in our test. And this printed list is a sample that will be used to train the SL model later. After being printed, the list is reset to empty and ready to store counters to generate the next sample. The entire duration to create a sample is called a test round. The controller repeats the above steps to generate new samples per test round, and each sample contains the variation of Packet_In messages. This is similar to the peak/off-peak hour monitor in the real network, in which the fluctuation of network usage in 24 h can be reflected in a list with 24 records when a time slot is 1 h. Since we

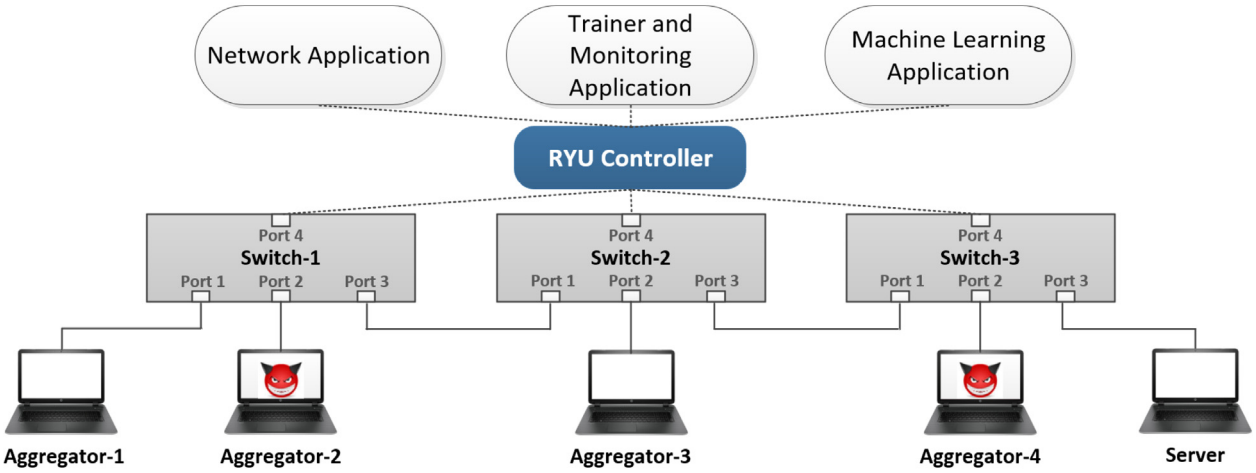


Fig. 2. Network topology used for results generation on mininet emulator.

can insert attack traffic in the test on purpose, the type of traffic, legitimate or malicious, is under control. Thus, we know how to label a sample, whether normal or attack, in the data preparation. After a group of normal and attack samples are obtained, we then remove duplicated records and randomly pick samples from the two categories, normal and attack, until we have enough samples in the training set. During sample selection, both the size of training set and the ratio of attack samples to the whole samples are considered. Once the training set is ready, it is sent to MLA for predictor training and evaluation. These labelled samples are divided into two parts, one for training (80%) and one for validation (20%). After a predictor being trained by 80% of the samples, it uses the rest 20% of the samples to verify whether its prediction is correct or not. MLA will create a confusion matrix, which is a table that describes the performance of a predictor, from these predictions, so that the administrator knows whether this predictor is trustworthy. If the outcome is satisfactory, this predictor can be deployed in the network for real time detection.

The objective of the offline MLA training is to allow the SL model to learn from a dataset the trends and dynamics of the network either in *normal* or *attack* scenarios. A single sample (D) in the training set records the continuous behaviour of a user device (k) per fixed duration (t), for n times. The selected network statistics are the number of Packet_In messages ($m_i(t)$) counted per MAC address ($i = 1, 2, \ldots, k$) in a time slot (t) as shown in (1):

$$m_i(t) = \sum_{1}^{t} Packet\_In(t) \tag{1}$$

In (2), normal (Dn) and attack (Da) samples are tagged with a class ID (label) that specifies the network state either *normal (class ID= no)* or *attack (class ID= yes)* for the training. The class ID is empty for real-time sample (Dr), as this needs to be predicted by the SL model.

$$\begin{aligned} D_n &= [m_1, m_2, m_3, \ldots, m_n, no] \\ D_a &= [m_1, m_2, m_3, \ldots, m_n, yes] \\ D_r &= [m_1, m_2, m_3, \ldots, m_n, -] \end{aligned} \tag{2}$$

An example of the training set in our test is described in Table 2. Each column (column 2 to 7) represents a sample, 3 normal samples (column 2 to 4) and 3 attack samples (column 5 to 7). Each sample consists of counters in 30 consecutive time slots (TS1 to TS30) and a label which indicates the network status. We can see that samples under normal scenarios (labelled as 'No') have some burst traffic whose counts exceed 100, or even 1000. This is to say that as long as the traffic is classified as normal in the training set, the predictor is able to distinguish this sudden burden from attack traffic. Comparing to threshold based detection, our proposed method allows burst traffic if it is learnt during training. After the training, MLA is ready for DDoS attack detection.

In the real time detection phase, similar to the training sample preparation phase, TMA collects statistics via the controller every t time, and stores this value to a list, after the list has enough records, which is 30 in our test, TMA sends this list to MLA for DDoS attack detection. Here the list is in the same format as the training sample. Then, MLA decides whether the network is in the *normal* or *attack* state based on the previous training. It provides a decision with a probability ranges from 0 to 1, by default normal and attack scenarios expect the value to be $0 \leqslant x < 0.5$ and $0.5 \leqslant x \leqslant 1$, respectively. It is worth noting that these two ranges are set by default, they can be modified, i.e. 0–0.7 for normal and 0.7–1 for attack, according to the validation result to gain a better prediction performance. To evaluate each SL technique in general, we will just use the default range in our test. MLA informs NA the prediction so that NA can run countermeasures to block DDoS attacks when the network state is abnormal. So far, the offline training phase is separate from the detection phase, because the real-time traffic cannot be classified for 100% accuracy, which could mislead MLA in the training.

4. Evaluation using experimental dataset

Before implementing in the SDN testbed, we first evaluate the accuracy of each SL techniques using some experimental datasets: 1999 DARPA [41], DDoS attack SDN dataset (DASD) [42] and InSDN [43]. 1999 DARPA is collected via the evaluation of IDS in a simulated network, it provides the time period of attack, as well as the attack type. Data in DARPA is recorded in the raw format in Wireshark, which is a network protocol analyser, and a timestamp for each packet is given. DASD is collected from Mininet, it contains benign traffic and DDoS attack traffic, such as UDP flooding, 22 features are available in this dataset. Traffic in DASD is labelled in two categories, DDoS attack and normal. InSDN is also an SDN based dataset, it contains attacks against both the control and data plane in a virtual environment. Comparing to DASD, InSDN records 83 features and classifies multiple attacks, including DDoS attacks and probe attacks. Furthermore, since it emulated flooding attacks against the SDN controller and the Packet_In messages are captured by Wireshark, we will use this dataset to evaluate our model later.

Table 2
Example of normal and attack training samples.

	Number of Packet_In messages received in a time slot					
TS1	969	1005	993	398	4	5
TS2	953	980	956	4	5	5
TS3	975	985	986	6	5	5
TS4	912	997	1003	5	5	644
TS5	986	989	980	5	1366	4
TS6	934	954	982	644	5	6
TS7	989	1000	992	5	5	4
TS8	964	990	972	5	5	5
TS9	933	1013	993	5	4	130
TS10	1002	1024	1010	4	1122	518
TS11	937	989	1009	534	162	5
TS12	1001	934	968	115	4	5
TS13	1035	998	984	4	6	4
TS14	1008	982	958	6	4	6
TS15	965	1015	991	5	6	642
TS16	977	992	1007	5	1186	5
TS17	1000	998	1008	643	4	6
TS18	969	959	976	4	6	4
TS19	968	995	967	6	4	6
TS20	970	981	962	5	5	643
TS21	979	996	965	5	1278	5
TS22	915	996	985	644	5	5
TS23	995	985	1015	5	5	5
TS24	948	981	973	4	5	5
TS25	964	970	1010	6	4	198
TS26	980	965	965	6	429	450
TS27	941	974	974	233	763	4
TS28	957	1013	1019	414	4	6
TS29	980	990	970	4	6	4
TS30	950	932	1001	5	4	6
Label	**Yes**	**Yes**	**Yes**	**No**	**No**	**No**

Prior to SDN implementation, we employ these three datasets to evaluate the model in the DoS/DDoS attack detection. Validation is accomplished via the features available in these datasets. Here, we only concern about the DoS/DDoS attack traffic and normal traffic in the dataset. Thus, we manually filter related samples from the dataset and pre-process to generate the training set. We prepare three groups of training sets which contain 5%, 25% and 50% attack samples by randomly selecting from a dataset, each group has 10 sets of data and each set has 32,000 samples. As DASD and InSDN have so many features, feature reduction is applied to investigate the performance when only key features are considered in training a model. The correlation value, ρ, between each feature and the classifier is measured, either a strong positive or negative correlation feature is regarded as a key feature. The choice of ρ is a trade-off between the number of features and the accuracy metric, because a large absolute value of ρ usually means fewer features and lower accuracy. As long as the user requirement for DDoS attack detection can be met, training predictors by a small number of features is preferred, as it saves system resources when processing, and simplifies data preparation.

In the DARPA, since each sample is packet-based, we choose the four default features, which are src_ip, dst_ip, protocol_type and length, in the Wireshark to compare its performance with various sizes of anomalous packet in the training set. It can be found in Table 3 that the accuracy reaches the highest value when having 25% attack packets in the training set. GLM, DT, KNN, and BT have stable outcomes regardless of the size of attack sample, however, 5% attack sample in the training set is not enough for SVM, NB, DA, and FNN to distinguish the DoS attack from normal data, especially NB and FNN. All the four metrics are presented in Fig. 3, as we will compare this result with our proposed model explained in Section 5. We can see from the figure that DA suffers from fake alarms, and the detection rate of SVM is unacceptable. Note that Fig. 3 is produced from training sets with 50% attack sample instead of 25% even though 25% attack sample has a slightly better accuracy. This is due to the consistency of other outcomes in this article. Moreover, the difference between using 25% and 50% abnormal records in training can be ignored.

In DASD, we find three key features, which are pktcount, bytecount and protocol, whose ρ values are greater than 0.25 or less than -0.25. Detailed comparison is shown in Table 4, the overall difference column lists the largest difference in accuracy when having different sizes of attack samples in the training set. Apart from accuracy, other metrics of DASD using all features and 50%

Table 3
Accuracy using 1999 DARPA.

ML Techniques	5% Attack	25% Attack	50% Attack	Difference
SVM	50%	56.17%	48.38%	7.79%
GLM	99.56%	99.67%	99.67%	0.11%
NB	75.26%	97.95%	97.42%	22.69%
DA	80.98%	85.46%	86.42%	5.44%
FNN	69.19%	94.74%	94.07%	25.55%
DT	99.39%	99.7%	99.57%	0.31%
KNN	99.39%	99.53%	99.4%	0.14%
BT	99.39%	99.69%	99.53%	0.3%

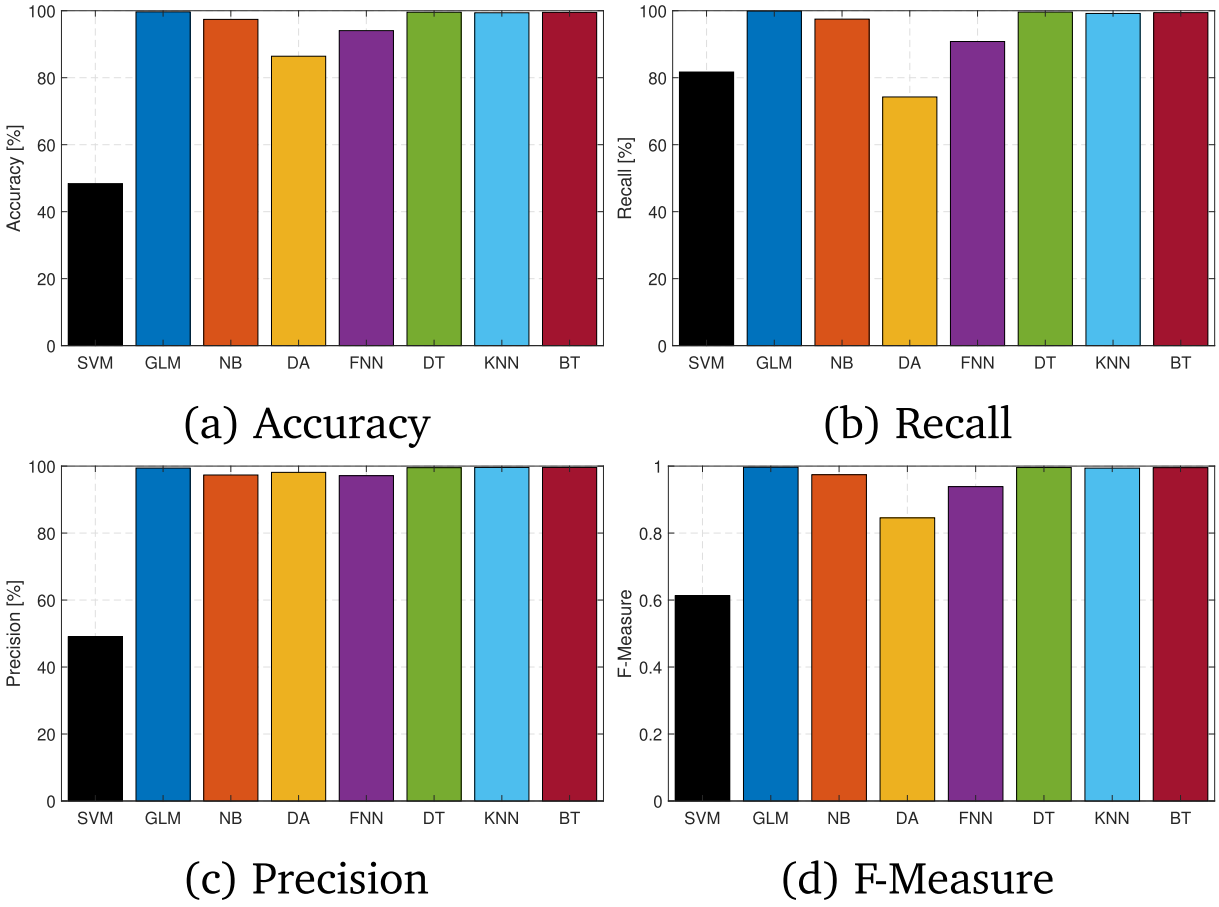


Fig. 3. Metrics for DoS attacks in 1999 DARPA with 4 default features and 50% attack samples in the training set.

Table 4
Accuracy using DASD.

ML	5% Attack		25% Attack		50% Attack		Overall Difference	
Techniques	All Features	3 Features	All Features	3 Features	All Features	3 Features	All Features	3 Features
SVM	49.42%	47.53%	49.25%	45.3%	49.47%	54.3%	0.22%	9%
GLM	53.19%	50%	63.35%	55.75%	73.82%	65.91%	20.63%	15.91%
NB	85.87%	69.2%	89.81%	78.69%	90.4%	80.43%	4.53%	11.23%
DA	71.24%	59.16%	76.17%	65.8%	85.3%	69.67%	14.06%	10.51%
FNN	45.2%	50.59%	44.84%	51%	44.32%	51.01%	0.88%	0.42%
DT	92.17%	90.52%	99.74%	96.59%	99.78%	97.78%	7.61%	7.26%
KNN	87.62%	92.11%	95.17%	97.68%	96.63%	98.63%	9.01%	6.52%
BT	98.19%	91.94%	99.94%	97.64%	99.83%	97.74%	1.75%	5.8%

attack samples are given in Fig. 4. From the output of SVM and FNN, the precision is lower than 60%, which means anomalous data have over 40% to be ignored, it is quite unreliable for DDoS attack detection.

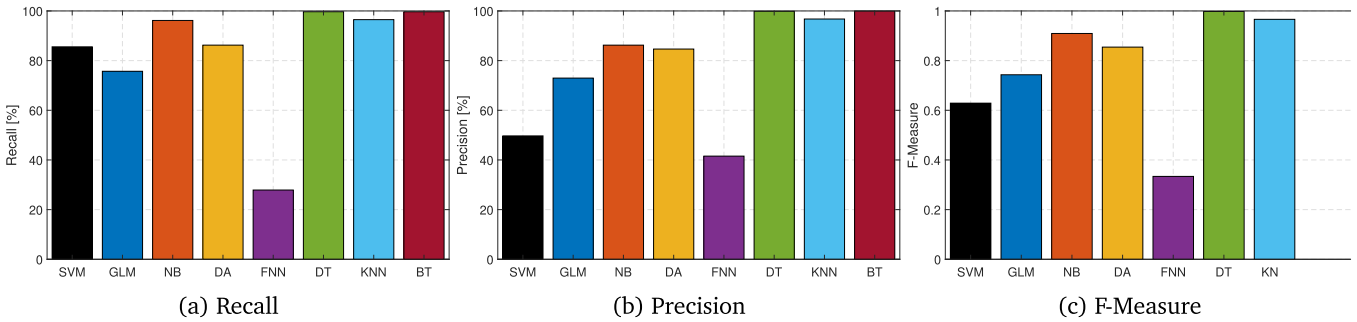


Fig. 4. Metrics for DoS/DDoS attacks in DASD with all features and 50% attack samples in the training set.

Since some features in the InSDN are useless in the DoS/DDoS attack detection, i.e. fwd_psh_flags is always 0 under both normal and attack scenarios, we decide to filter those irrelevant features first rather than use all of them, and then choose the essential features from InSDN to compare the difference. Thus, we regard i.) a feature as irrelevant when its absolute ρ value is less than 0.1, and this returns 38 features; ii.) a feature as vital when its absolute ρ value is greater than 0.5, and this returns 7 features which are flow_id, protocol, timestamp, flow_pkts/s, bwd_pkts/s, pkt_len_min and init_bwd_win_byts. The choice of ρ takes the number of features into consideration, as we would like to have a large group of relevant features and a small group of crucial features. The accuracy is shown in Table 5, and all the four metrics are given in Fig. 5, as we will compare with our proposed model in Section 5.

From Tables 4 and 5, similar to the results in the DARPA, the impact from changing the proportion of attack data in the training set can be ignored for most of the techniques. If we look into the result under the same group of dataset, considering more features in the training phase could lead to two opposite outcomes: a better accuracy or overfitting. For most SL algorithms, having more features in the training gives slightly better performance; while, FNN in the DASD and KNN in the InSDN have been misled by non-essential features, their accuracy degrades when taking more features into consideration.

KNN and BT have excellent and solid accuracy in all the three datasets, while SVM is quite unreliable. From the comparison of accuracy using the above three datasets, we can conclude that:

- Using more features in the training usually returns a better accuracy, although it leads to overfitting for certain models.
- Filter and employ only high correlated features could be a feasible option, accuracy is more or less sacrificed for a compact version of training set, which results in an easier collection of data and less resource consumption.

Table 5
Accuracy using InSDN.

ML	5% Attack		25% Attack		50% Attack		Overall Difference	
Techniques	38 Features	7 Features	38 Features	7 Features	38 Features	7 Features	38 Features	7 Features
SVM	50%	50%	50.8%	53.44%	50.3%	67.57%	0.8%	17.57%
GLM	99.48%	98.02%	99.84%	98.78%	99.96%	98.88%	0.48%	0.86%
NB	81.35%	99.58%	83.39%	99.5%	84.02%	99.28%	2.67%	0.3%
DA	98.84%	99.46%	99.07%	99.54%	99.11%	99.64%	0.27%	0.18%
FNN	49.48%	48.33%	49.92%	48.4%	49.92%	45.28%	0.44%	3.12%
DT	84.4%	99.01%	76.09%	99.92%	67.05%	71.57%	17.35%	28.35%
KNN	99.97%	99.83%	99.94%	99.99%	99.97%	99.99%	0.03%	0.16%
BT	99.77%	99.04%	99.99%	99.83%	99.95%	97.86%	0.22%	0.82%

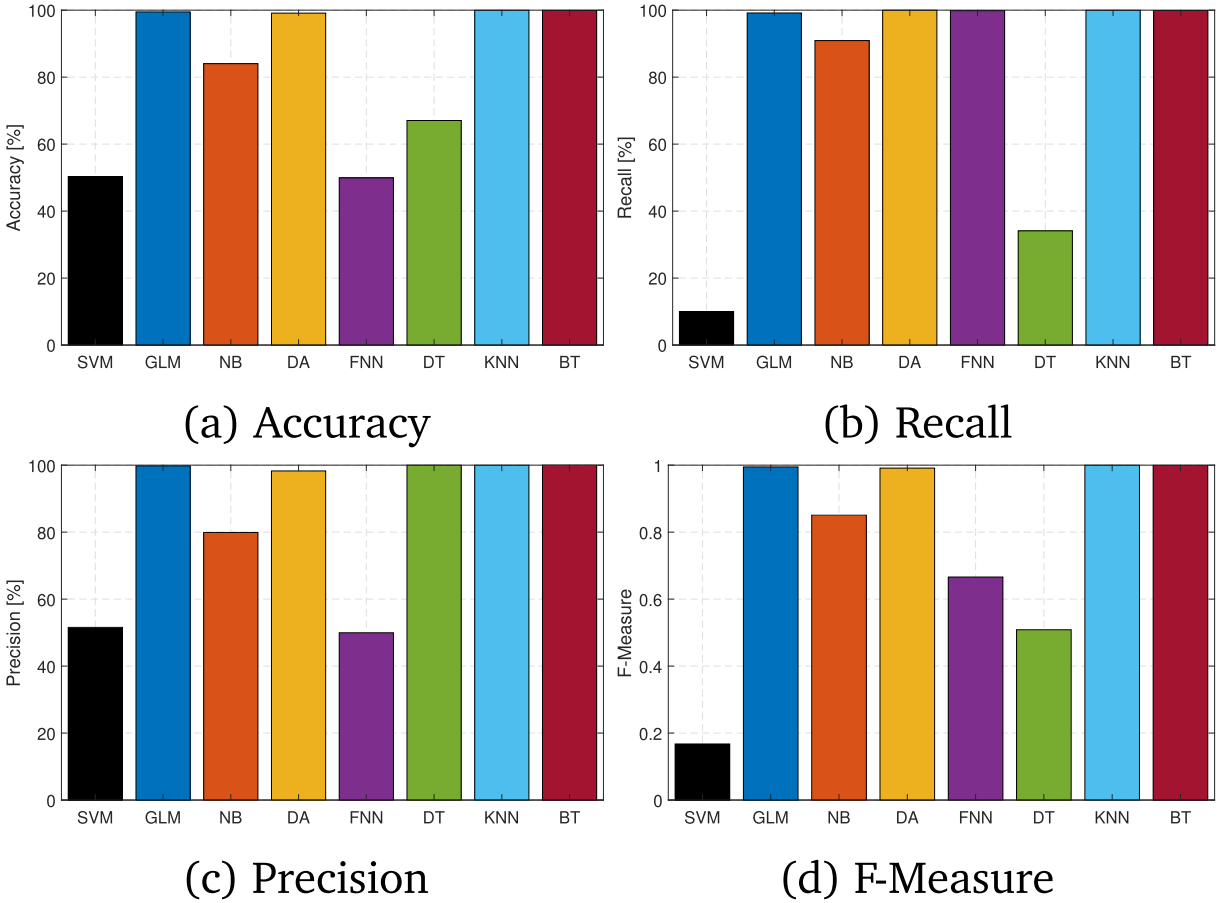


Fig. 5. Metrics for DoS/DDoS attacks in InSDN with 38 features and 50% attack samples in the training set.

- The difference of accuracy between 25% and 50% attack sample in the training set is small when the accuracy is over 90%.
- Having more attack samples in the training set generally results in a slightly higher accuracy when the number of features involved is large.
- Accuracy indicates an overall performance of SL algorithms, but issues may be hid under extreme situations. Thus, using multiple metrics together to evaluate the performance is a more reliable way.
- A suitable SL algorithm needs to be explored before implementation, because the performance of some SL techniques could be poor in certain scenarios.

5. Experimental results in the network emulator and real testbed

In this section, we discuss the details on how the training and prediction are carried out, as well as the outcomes of our proposed model.

5.1. Training sample generation

The training set is obtained from Mininet testbed. In our test, we define a training sample to be the statistics in 300 s, and this 300-s period is divided into 30 time slots, $n = 30$, each time slot has 10 s duration, $t = 10$. These parameters can be varied depending on the use cases in the real network, Since new packets received on the switch will trigger Packet_In requests to the control plane, the RYU controller records the number of Packet_In messages per 10 s in a list until the list has 30 records, and then this list is printed out and reset to empty, so that it is ready for the next test round. Note that each sample is composed of 31 variables (30 numbers and 1 class ID) after we manually classify the sample.

The network statistics required in our test is only the number of Packet_In messages received on the controller per aggregator/server, $k = 5$, within each time slot. For a normal scenario, traffic is generated with periodical and random TCP/UDP scenarios as mentioned before. For an attack scenario, there are both normal traffic and attack traffic in the network at the same time. However, the controller only counts the number of incoming requests regardless of the feature inside messages. In order to verify the relationship between the number of samples used for training and the accuracy of the detection, a numbers of datasets are prepared (125, 250, 500, 1000, 2000, 4000, 8000, 16,000, and 32,000). The main purpose is to compare the difference in output for each SL technique using different size of training sets. In each size of training set, both normal and attack samples are 50% to ensure the training phase has no bias. In the group of 32,000 records, we also verify the performance with a lower proportion (5% and 25%) of attack samples in the training set, because the ratio of anomalous traffic to legitimate traffic is usually low in the real world.

5.2. SL training

To validate the proposed model, we use holdout validation. In each training set, 80% samples are used for training and 20% reserved for validation [44], i.e. 25,600 samples for training and 6400 samples for validation when 32,000 data in total. Out of the validation, a confusion matrix is created after training, and further performance evaluation, such as accuracy, is calculated based on this matrix. To provide accurate statistics regarding the accuracy of SL, experiments of both training and validation are performed 10 times using datasets with different sample sizes (random allocation is used for picking training and validation samples).

Since a SL model could be deployed on the resource-constrained device, the resource consumption shall also be considered while selecting a SL technique. Moreover, how quickly a SL model can make a prediction is another key factor. To compare the resource consumption and response time, each SL technique is trained by the same training set (25,600 samples) to measure the training period and CPU utilisation. And the duration from when a sample is sent to the model till the prediction is made is defined as the detection time. Since the training and prediction are executed in Matlab, we put tic and toc functions before and after the training/prediction operation in the Matlab, respectively. The tic and toc functions in the Matlab record the duration of operations between them. Thus, we can measure the training and detection time of each SL technique. Each test runs 10 times to obtain the mean result as shown in Tables 6. The result shows that BT and SVM need more time for training than others, taking 40 s and 26 s, respectively. Considering that most of the SL techniques only spend less than 1 s, these two algorithms require a longer initialisation time. Their CPU consumption is over 50%, making them the top two heavy users among all the SL techniques. Nevertheless, KNN and NB are the two fastest and most CPU efficient learners, taking less than 0.3 s for training along with around 7% CPU utilisation. As the training is an independent process running at the beginning of network initialisation in the proposed model, the consumption in the system will not impact the performance of DDoS attack detection. However, the decision time of each technique is related to the detection time in the network monitoring. From the result, we can observe that DT costs only about 0.003 s, which is the least time to make a decision. In comparison, BT needs more than 100 times longer than DT to predict the state.

5.3. SL detection results

With the metrics defined at the beginning of Section 2, the mean successful prediction rate for both normal and attack states are shown in Fig. 6 with 95% confidence interval. For most of the SL techniques we analysed, the fluctuation of output shrinks as the size of training set increases, which means the prediction becomes stable with enough training. Some techniques, such as KNN and BT, have a trend to reach 100% accuracy with a large training set. While others, such as SVM, do not change significantly in the output. This means the size of a training set usually influences the performance of prediction. Note that not all the SL techniques are adaptive to our training method. Also, the accuracy, precision, and F-measure of SL techniques with different training size are given in Fig. 7. To be more specific, Fig. 8 shows the accuracy, recall, precision, and F-measure of SL techniques with a training set of 32000. As we can see, DT, KNN, and BT are the three best techniques following the way we train. BT has an accuracy of 99.46%, which is the best among the eight SL techniques we considered.

The outputs of some SL techniques, such as KNN and BT, have excellent performance with only one feature picked for training,

Table 6
Resource consumption of SL techniques.

SL Technique	Training Time (s)	CPU Utilisation (%)	Decision Time (s)
SVM	25.583484	50	0.0056511
GLM	0.7646198	12.8	0.0040819
NB	0.2963912	7	0.0050104
DA	0.4379895	8	0.0100427
FNN	2.3837825	41.9	0.009235
DT	0.6213096	11.3	0.002857
KNN	0.2741382	5.8	0.0050494
BT	40.1033767	52.1	0.3169704

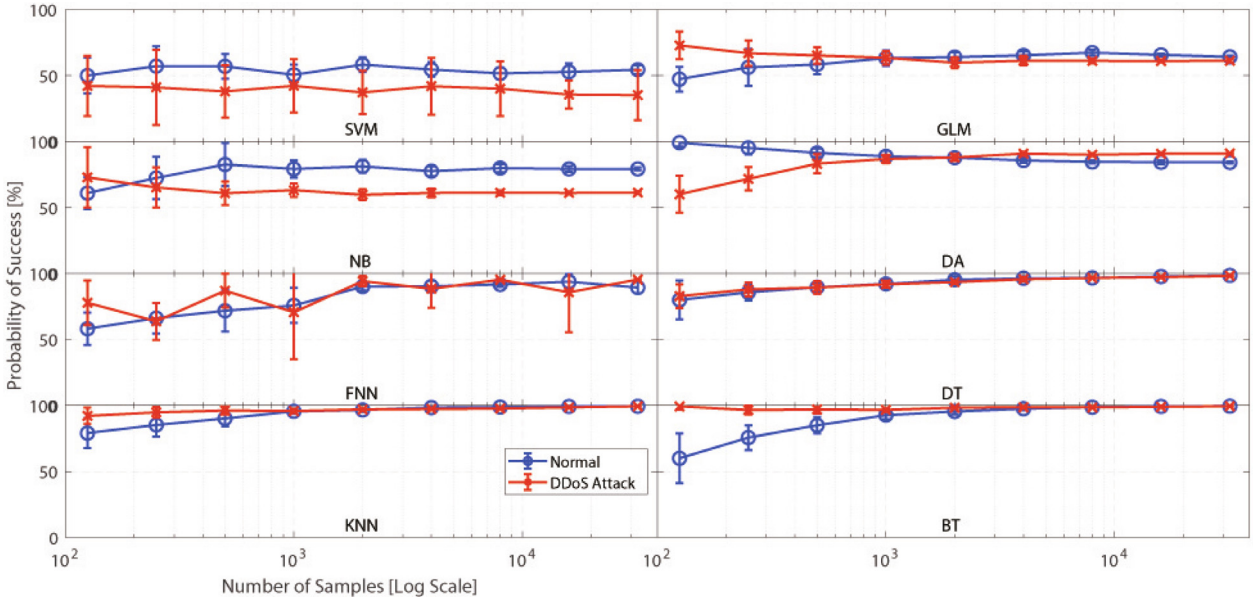
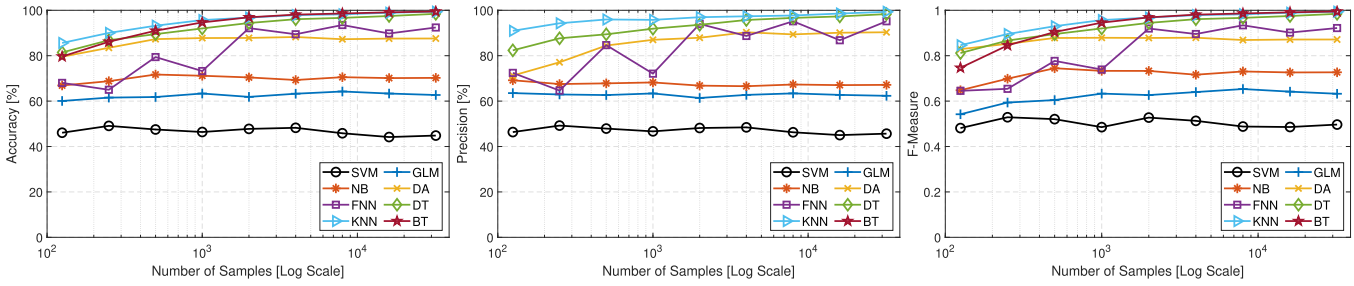
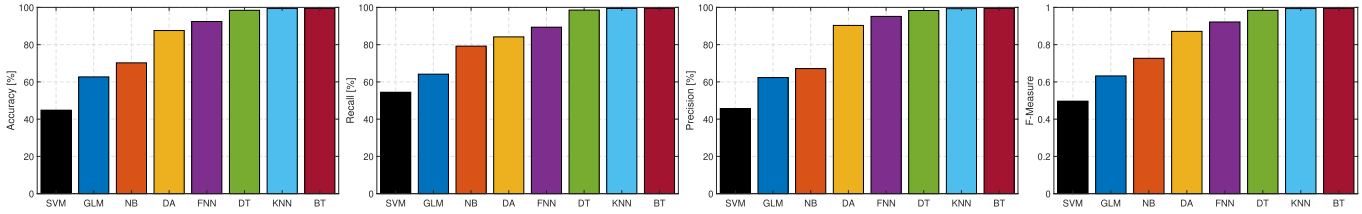


Fig. 6. Normal and DDoS attack scenarios using different SL techniques for DDoS attack detection.



(a) Accuracy (b) Precision (c) F-Measure

Fig. 7. Metrics for assessing different SL techniques for detecting DDoS attacks in SDN.



(a) Accuracy (b) Recall (c) Precision (d) F-Measure

Fig. 8. Metrics for assessing different SL techniques in detecting DDoS attacks with 32000 samples.

which means it is able to apply simple feature to distinguish abnormal behaviours from normal traffic. Thus, there will be several advantages from this outcome: (i) the collection of training sets becomes easy since there is no need to probe details of traffic flows; (ii) training phase becomes simple as there is no need to define critical features; (iii) resource consumption in training is reduced due to less complexity of the training set.

5.4. SL parameters modification

Parameter change in the SL technique could lead to performance improvement. Using different models has a significant impact on the output. This is because of the features in the training set, as well as the way the model is trained. Here, we scan possible values in each technique, and find four of them are improved more or less. In Tables 7 and 8, we list the four techniques that have performance growth after adjusting the training parameters. Among the four algorithms, GLM has the biggest improvement that is enhanced up to 56%.

For each SL technique, the parameters changed are listed:

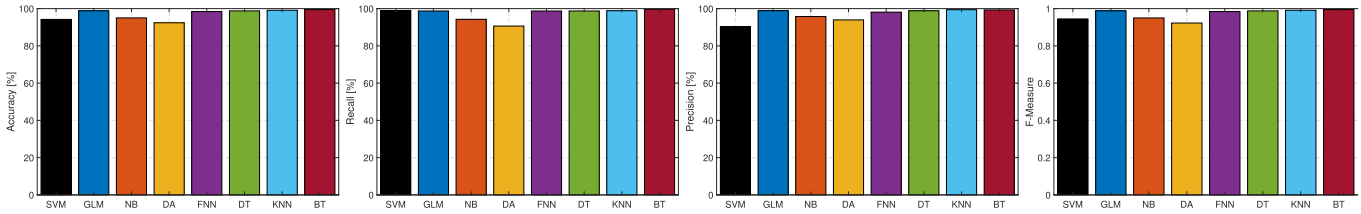
- For FNN, we use 5 hidden layers in the original test. Then, we increase the layer from 5 to 10. In doing so, we find that with 9 hidden layers, the output is the best.
- For GLM, we set the model specification from "linear" to "quadratic" mode, as well as the link function from "logit" to "comploglog". Consequently, its performance improves a lot. This is because the squared term for the predictor is much closer to

Table 7
Accuracy improvement from parameter change.

SL Technique	Samples	Original (%)	New (%)	Growth (%)
FNN	8000	93.5124	96.3745	3.1
	16000	89.793	96.4395	7.4
	32000	92.4221	97.7025	5.7
GLM	8000	64.2366	95.8308	49.2
	16000	63.3317	97.4939	53.9
	32000	62.6908	97.9035	56.2
KNN	8000	98.2262	98.4073	0.2
	16000	98.9147	99.002	0.1
	32000	99.3731	99.4261	0.1
DT	8000	96.6627	96.7941	0.1
	16000	97.4808	97.6092	0.1
	32000	98.4398	98.5563	0.1

Table 8
F-Measure improvement from parameter change.

SL Technique	Samples	Original (%)	New (%)	Growth (%)
FNN	8000	1.8837	1.9009	0.9
	16000	1.7204	1.9019	10.5
	32000	1.8834	1.9281	2.4
GLM	8000	1.2562	1.8746	49.2
	16000	1.2429	1.9092	53.6
	32000	1.2346	1.9171	55.3
KNN	8000	1.935	1.9388	0.2
	16000	1.9519	1.9524	0.03
	32000	1.9668	1.9674	0.03
DT	8000	1.9138	1.9163	0.1
	16000	1.9268	1.9319	0.3
	32000	1.947	1.9516	0.2



(a) Accuracy (b) Recall (c) Precision (d) F-Measure

Fig. 9. Metrics for medium traffic with 32000 samples.

our training set than the linear mode. A double-log link function better reflects the relationship between the mean response and predictors.

- For KNN, we change the distance mode from standardized euclidean to city block and observe a bit improvement.
- For DT, we try MinParentSize from 2 to 10 to control tree depth. As a result, set it to 5 has the best outcome.

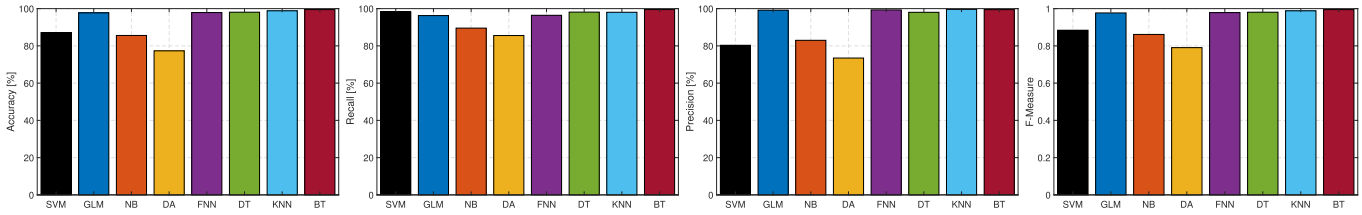
5.5. Verify different traffic volumes with modified parameters

Since the previous results are given by low volume traffic, *i.e.*, 100 bps UDP traffic, we also verify the performance of SL in medium and high volume traffic. Due to the limitation of available bandwidth, which is less than 90Mbps, in the testbed, we define 1Mbps and 10Mbps UDP traffic as medium and high volume scenarios, respectively. Note that the TCP traffic always occupies the entire available bandwidth, which is to say if TCP and UDP traffic transmits simultaneously, the bandwidth of TCP traffic will be the whole bandwidth subtracts the bandwidth used by UDP traffic.

We repeat the analytic in Fig. 8 with medium and high traffic volume. The test uses the parameters discussed in Section 5.4 to get the best output for each SL technique. From the output in Figs. 9 and 10 can be seen that the performance of SL techniques, such as SVM and DA, has remarkably improved than under low traffic. While the performance of BT is still the best among all the SL techniques: the recall reaches 99.7% under medium traffic and 99.49% under high traffic. Thus, the model we proposed in this article can be deployed to analyse traffic in general.

5.6. Implementation on the real testbed

To verify the performance of SL techniques on the real testbed, we choose the top four performed SL techniques: FNN, DT, KNN, and BT, which have over 90% accuracy when trained and validated with 32,000 samples. The topology is the same as in Fig. 2 with Zodiac switches and Raspberry Pis working in the physical testbed. Statistics of the real testbed are collected in the same way as in the simulation to generate a training set, so that validation is done all through the physical network from training to prediction. Each SL



(a) Accuracy (b) Recall (c) Precision (d) F-Measure

Fig. 10. Metrics for high traffic with 32000 samples.

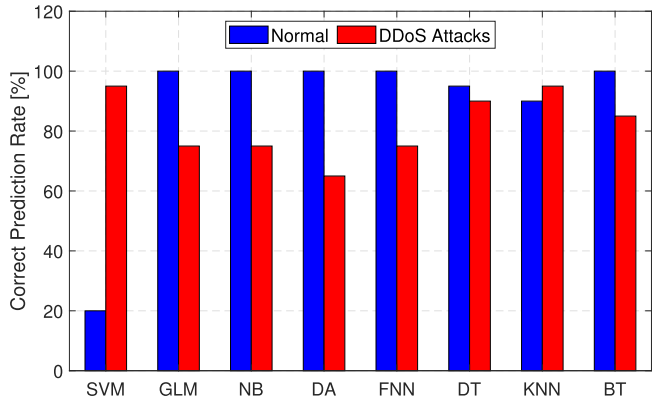


Fig. 11. True positive (normal) and true negative (attack) rate on the real testbed.

model is trained before routing policy is available in the network, and then the predictor judges every incoming sample (Dr) with a number "0" or "1" as the decision that is explained later. Also, we set parameters to the ones that can maximise performance as given in Section 5.4.

The real-time output over physical testbed is given below:

- *FNN*: The output of FNN is a value between 0 and 1, which gives the probability of the network state. If it is greater than 0.5, FNN considers the network is under attack; otherwise, it is normal.
- *DT*: As there are only two options in our test, DT defines the normal state as branch 1 and the attack state as branch 2, so the output is straightforward.
- *KNN*: KNN also prepares two groups of outputs that group 1 and 2 are normal and attack state, respectively.
- *BT*: BT casts vote under each state and larger value wins. The left one represents the normal state, while the right one represents the attack state.

In order to verify the performance over real-time traffic, we choose 20 consecutive decisions made by each SL model (trained by low traffic) under two network states to see the accuracy in Fig. 11. We can see from the output that most of the predictions in the normal state is better than in the attack scenario, especially with GLM, NB, DA, FNN, and BT, it reaches 100% in predicting normal state. The accuracy of DT, KNN, and BT are 92.5%, while correct prediction rate of DT and KNN are all over 90%.

5.7. SL accuracy comparison

As the previous tests use 50% of attack samples in the training set, in order to compare the accuracy with different proportion of attack samples, we create training sets who have 32,000 records with 5% and 25% attack samples. The outcomes of low, medium and high volume traffic are displayed in Table 9. Each accuracy value in the table is the mean value of 10 groups of training sets. DT, KNN, and BT are the three best performed algorithms in the proposed model, and the size of attack samples in the training

Table 9
Accuracy of proposed methods with various ratios of attack data in the training set.

	SL Technique	5% Attack	25% Attack	50% Attack	Difference
Low	SVM	42.8%	39.45%	49.09%	9.64%
	GLM	49.93%	77.53%	97.9%	47.97%
	NB	76.13%	75.62%	71.7%	4.43%
	DA	64.96%	64.3%	87.58%	23.28%
	FNN	68.34%	83.6%	97.7%	29.36%
	DT	95.63%	98.29%	98.56%	2.93%
	KNN	97.69%	99.02%	99.43%	1.74%
	BT	97.91%	99.27%	99.46%	1.55%
Medium	SVM	96.53%	95.42%	94.21%	2.32%
	GLM	97.34%	98.14%	98.83%	0.69%
	NB	95%	95.01%	95.03%	0.03%
	DA	92.59%	92.21%	92.4%	0.19%
	FNN	97.31%	97.75%	98.41%	1.1%
	DT	97.53%	98.73%	98.79%	1.26%
	KNN	99.38%	99.34%	99.12%	0.26%
	BT	99.4%	99.64%	99.51%	0.24%
High	SVM	84.12%	86.03%	87.11%	2.99%
	GLM	88.06%	94.67%	97.72%	9.66%
	NB	85.45%	85.33%	85.6%	0.27%
	DA	75.91%	76.67%	77.33%	1.42%
	FNN	88.85%	95.27%	97.88%	9.03%
	DT	95.49%	97.84%	98.08%	2.59%
	KNN	98.74%	99.02%	98.83%	0.28%
	BT	98.3%	99.26%	99.47%	1.17%

set has limited impact on the accuracy. For most SL algorithms, more attack samples lead to higher accuracy.

Since the DARPA dataset has timestamp, we generate a training set with the number of packets according to the record in DARPA, in order that SL algorithms are trained in our proposed method. We choose the flooding DoS attack, whose type is called 'neptune', in the DARPA, and count the number of packets towards the victim per second (t = 1). Then, we put the record in 30 s as one sample, so that each sample has 30 counts (n = 30), which is the same as our self-generated traffic. The reason that we only count per second rather than 10 s is the lack of attack samples in the DARPA. The size of this training set is 600 with 50% attack data. Similarly, we filter the Packet_In messages from the Wireshark records in the InSDN and create a training set that only contains the fluctuation of Packet_In messages, because our model considers DDoS attacks against the controller using the count of Packet_In messages. Due to the lack of attack samples, we set t = 1 and n = 10 to obtain a small training set with 300 samples, and the ratio of attack sample is also 50%. Note that we still generate 10 groups of training set by randomly choosing from the sample pools from DARPA and InSDN. Although the size of training set is relatively small comparing to our self-generated training set, the outputs from DARPA in Fig. 12 and from InSDN in Fig. 13 are worth further investigation. On one hand, most of the SL algorithms, especially KNN and BT, have convincible results in the DDoS attack prediction, but on the other hand, some algorithms are unstable, such as GLM and NB. SVM performs well in DARPA, however, it is totally lost in InSDN dataset, it predicts all the traffic as malicious so that the accuracy is 50% while other metrics are all 0. Since we define normal state as positive in our method, the true positive rate becomes 0, which causes recall, precision and F-measure to be 0. Although its response to any test set is always negative, it fortunately has 100% correction in the true negative rate, which makes the accuracy be 50%. From the implementation so far, DT, KNN and BT are the top three algorithms that are suitable to our model. When we compare Fig. 12 with Fig. 3, and compare Fig. 13 with Fig. 5, those well performed techniques, such as KNN and BT, still have remarkable performance, moreover, the overall performance is improved for most algorithms, and our proposed model only needs one feature.

In Table 10, we summarise the accuracy of related SL techniques, and compare with the performance in our simulation and physical test under low volume traffic. Because the low volume scenario has inferior overall performance to medium and high traffic, so that it could be the worst-case scenario of our model. It is worth noting that the performance of each work is based on a different dataset with distinctive features, so the comparison is to reflect the best output under a specific training strategy. As we can see, the best accuracy from literature review is 99.1% in [13], while BT has 99.46% in our test. Moreover, the top three ones in terms of accuracy in the simulation are DT, KNN, and BT, all above 98%. This is because these three algorithms suit our training method better than others. Also, this is reflected in Fig. 6, recall metrics of DT, KNN, and BT become stable as the number of training sets increases. However, other techniques still have fluctuation even trained by a large number of datasets. Although the dataset

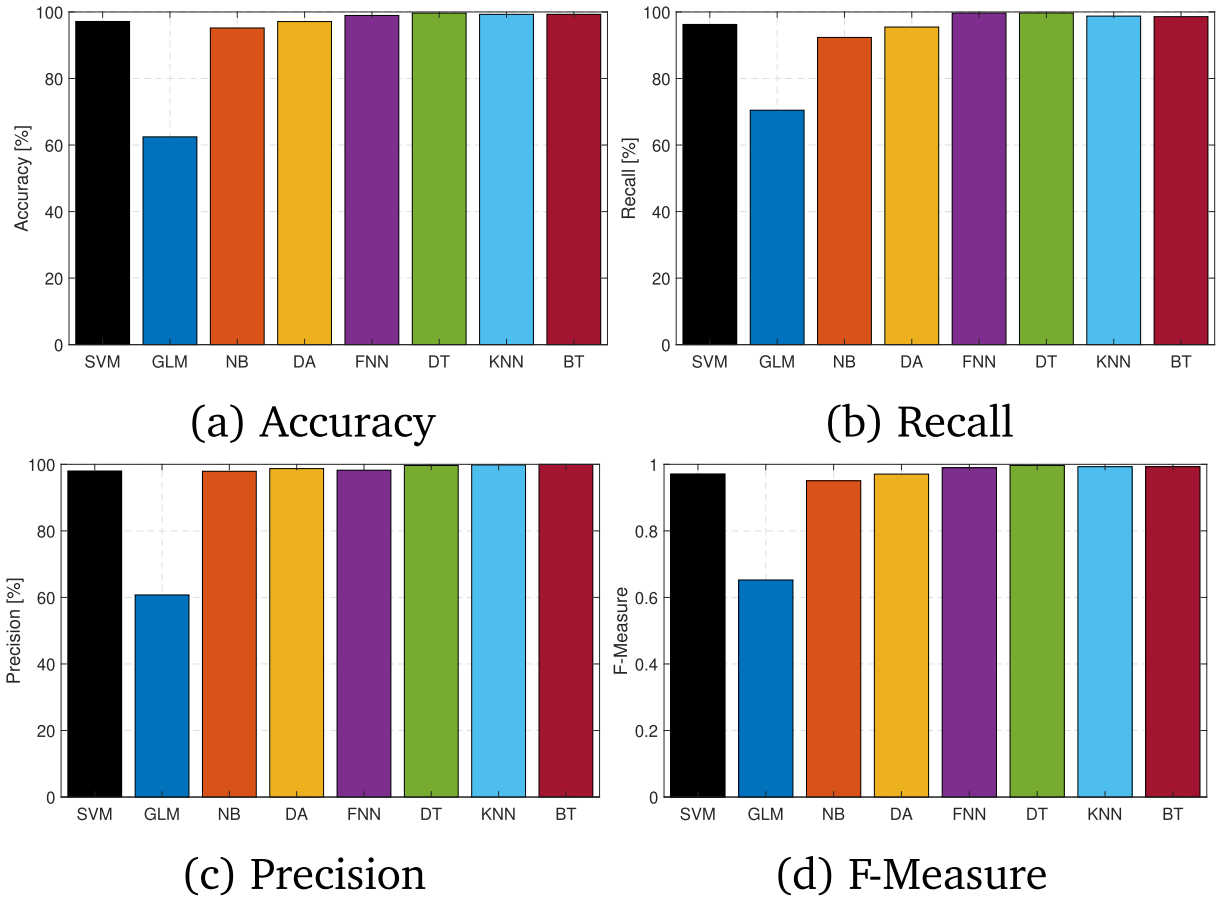


Fig. 12. Metrics for DoS attacks in 1999 DARPA using the proposed model.

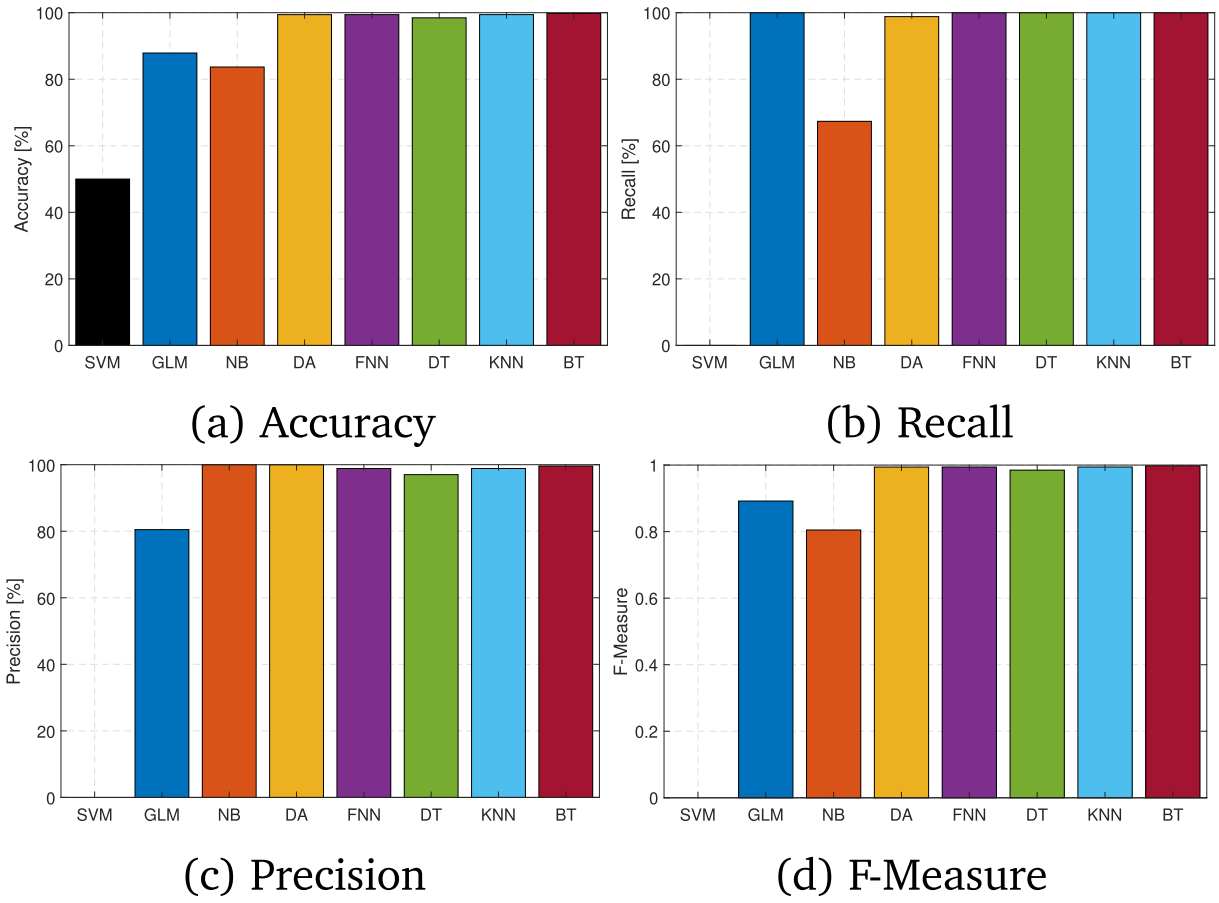


Fig. 13. Metrics for DoS/DDoS attacks in InSDN using the proposed model.

Table 10
A comparative analysis of SL accuracy comparison.

SL Technique	Meti et al. [11]	Zekri et al. [12]	Tuan et al. [13]	Sahoo et al. [14]	Bakker et al. [16]	Polat et al. [17]	Niyaz et al. [23]	Wang et al. [24]	Our Emulation	Our Real-Test
SVM	80%	**X**	**X**	98.9%	93.47%	92.46%	**X**	97.6%	49.09%	57.5%
GLM	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	97.9%	87.5%
NB	70%	91.4%	**X**	**X**	**X**	95.7%	**X**	**X**	71.7%	87.5%
DA	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	87.58%	82.5%
NN	80%	**X**	98.8%	**X**	**X**	92.28%	95.23%	**X**	97.7%	87.5%
DT	**X**	98.8%	98.2%	**X**	**X**	**X**	**X**	**X**	98.56%	92.5%
KNN	**X**	**X**	99.1%	92%	92.58%	98.3%	**X**	**X**	99.43%	92.5%
BT	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	99.46%	92.5%

and test environment are different, the performance by training via a simple feature still achieves a great accuracy compared to the results from the literature.

6. Discussion and future directions

In this article, we study and employ SL techniques to assist in DDoS attack detection over SDN. We explore the possibility to detect DDoS attacks from a single feature which is the counter within a time period. Training set is collected from experimental datasets and emulated traffic in the SDN. Different types of training sets have been tested to see the impact on output. To compare the performance of each SL technique, metrics in the detection, as well as CPU consumption and response time have been validated. Simulation results show that the performance of each SL technique varies far from each other under the same test scenario, while a modification in the parameter could improve the detection output. We conclude that bagging tree is the best SL technique in our proposed model, its accuracy reaches 99.64%. The real implementation shows that a real-time DDoS attack detection using SL in SDN has a satisfactory accuracy. In general, since SL requires a large amount of labelled data to train the predictor, it is able to detect various types of DDoS attacks with enough dataset. As we have discussed in the related works, the performance of a SL model using the same dataset differs from diverse SL techniques, thus, it is better to evaluate the prediction among candidate SL techniques to find the best one before deployment. The limitation of our proposed framework is that it is only suitable for flooding DDoS attacks, it is unable to detect non-volumetric attacks, such as low-rate DDoS attacks, and it only considers attacks against the control plane. In the future, we plan to investigate spoofed DDoS attacks with real SDN traffic using ML, and it will be better to achieve real-time

training so that the system is always up-to-date. Also, using the gradient of the number of packets within a monitor window could be another detection method with a single feature.

7. Conclusions

SDN redefines network management and communications, thus leading to network reforms. Although there are a number of benefits, SDN poses new security challenges, which requires attention of both academia and industry. The target of DDoS attacks is not only the device in the data plane, but also the control plane in the SDN. Due to the complexity in the features of traffic flow nowadays, the detection of DDoS attacks becomes more arduous than in the traditional network. With the help of the decoupled architecture of SDN, we are able to build a new model to define DDoS attacks in a flexible way. Nevertheless, the behaviour of DDoS attacks in the SDN shall be analysed under new environments, so that the attack definition is reliable. ML, as a new trend in helping people make decisions across many areas, could be a novel solution that can work with SDN. From historical training sets, a SL classifier predicts the network state in real time and informs the controller. As high quality dataset is hard to obtain, we proposed a lightweight SDN based ML model to detect DDoS attacks to alleviate this issue. Our proposed model collects the number of Packet_In requests through the SDN controller per time slot, and analyses the fluctuation of flows to detect DDoS attacks against SDN controller. The ML procedures from data preparation to DDoS attack detection can all be done over a single controller, thanks to the centralised control of SDN. In our test, a SL model can be trained within 1 s, make a prediction under several milliseconds, and the real time detection accuracy in the SDN reaches over 90%, which is a remarkable outcome considering the single feature involved in the training. And this simplifies the data preparation in the ML training phase. Although the ratio of attack traffic to the entire traffic is low in the real network, it is suggested to include more attack samples in the training set than the real proportion to gain a better accuracy in the prediction.

CRediT authorship contribution statement

Song Wang: Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing – original draft. **Juan Fernando Balarezo:** Methodology, Validation, Writing – review & editing. **Karina Gomez Chavez:** Methodology, Writing – review & editing. **Akram Al-Hourani:** Methodology, Writing – review & editing. **Sithamparanathan Kandeepan:** Methodology, Writing – review & editing. **Muhammad Rizwan Asghar:** Writing – review & editing. **Giovanni Russello:** Writing – review & editing.

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.

Acknowledgment

Song Wang and Juan Balarezo are supported by the Australian Government Research Training Program Scholarship.

References

[1] F.X. Wibowo, M.A. Gregory, K. Ahmed, K.M. Gomez, Multi-domain software defined networking: Research status and challenges, J. Network Comput. Appl. 87..
[2] W. Xia, Y. Wen, C.H. Foh, D. Niyato, H. Xie, A survey on software-defined networking, IEEE Commun. Surveys Tutor. 17 (1) (2014) 27–51.
[3] B.A.A. Nunes, M. Mendonca, X.N. Nguyen, K. Obraczka, T. Turletti, A survey of software-defined networking: Past, present, and future of programmable networks, IEEE Commun. Surveys Tutor. 16 (3) (2014) 1617–1634.
[4] Q. Yan, F.R. Yu, Q. Gong, J. Li, Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges, IEEE Commun. Surveys Tutor. 18 (2015) 602–622.
[5] D. Kreutz, F.M. Ramos, P. Verissimo, C.E. Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking: A comprehensive survey, Proc. IEEE 103 (1) (2015) 14–76.
[6] A. Lara, B. Ramamurthy, OpenSec: Policy-based security using software-defined networking, IEEE Trans. Network Service Manage. 13 (1) (2016) 30–42.
[7] D. Yin, L. Zhang, K. Yang, A DDoS attack detection and mitigation with software-defined Internet of Things framework, IEEE Access 6 (2018) 24694–24705.
[8] S.T. Zargar, J. Joshi, D. Tipper, A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks, IEEE Commun. Surveys Tutor. 15 (2013) 2046–2069.
[9] B. Karan, D. Narayan, P. Hiremath, Detection of ddos attacks in software defined networks, in: 2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS), IEEE, 2018, pp. 265–270.
[10] D. Comaneci, C. Dobre, Securing networks using sdn and machine learning, in: 2018 IEEE International Conference on Computational Science and Engineering (CSE), IEEE, 2018, pp. 194–200.
[11] N. Meti, D. Narayan, V. Baligar, Detection of distributed denial of service attacks using machine learning algorithms in software defined networks, IEEE Conference on Advances in Computing, Communications and Informatics (2017) 1366–1371.
[12] M. Zekri, S. El Kafhali, N. Aboutabit, Y. Saadi, DDoS attack detection using machine learning techniques in cloud computing environments, IEEE Conference of Cloud Computing Technologies and Applications (2017) 1–7.
[13] N.N. Tuan, P.H. Hung, N.D. Nghia, N.V. Tho, T.V. Phan, N.H. Thanh, A ddos attack mitigation scheme in isp networks using machine learning based on sdn, Electronics 9 (3) (2020) 413.
[14] K.S. Sahoo, B.K. Tripathy, K. Naik, S. Ramasubbareddy, B. Balusamy, M. Khari, D. Burgos, An evolutionary svm model for ddos attack detection in software defined networks, IEEE Access 8 (2020) 132502–132513.
[15] M. Alkasassbeh, G. Al-Naymat, A. Hassanat, M. Almseidin, Detecting distributed denial of service attacks using data mining techniques, Int. J. Adv. Comput. Sci. Appl. 7 (1) (2016) 436–445.
[16] J.N. Bakker, B. Ng, W.K. Seah, Can machine learning techniques be effectively used in real networks against DDoS attacks?, IEEE Conference on Computer Communication and Networks (2018)
[17] H. Polat, O. Polat, A. Cetin, Detecting ddos attacks in software-defined networks through feature selection methods and machine learning models, Sustainability 12 (3) (2020) 1035.
[18] J. Huyn, A scalable real-time framework for DDoS traffic monitoring and characterization, in: IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, 2017, pp. 265–266..
[19] M.E. Ahmed, H. Kim, M. Park, Mitigating DNS query-based DDoS attacks with machine learning on software-defined networking, IEEE Military Communications Conference (2017) 11–16.
[20] A. Shiravi, H. Shiravi, M. Tavallaee, A.A. Ghorbani, Toward developing a systematic approach to generate benchmark datasets for intrusion detection, Comput. Secur. 31 (2012) 357–374.
[21] S. Dong, M. Sarem, Ddos attack detection method based on improved knn with the degree of ddos attack in software-defined networks, IEEE Access 8 (2019) 5039–5048.
[22] S.S. Mohammed, R. Hussain, O. Senko, B. Bimaganbetov, J. Lee, F. Hussain, C.A. Kerrache, E. Barka, M.Z.A. Bhuiyan, A new machine learning-based collaborative DDoS mitigation mechanism in software-defined network, in: IEEE Conference on Wireless and Mobile Computing, Networking and Communications, 2018.
[23] Q. Niyaz, W. Sun, A.Y. Javaid, A deep learning based DDoS detection system in software-defined networking (SDN), arXiv preprint arXiv:1611.07400..
[24] P. Wang, K.M. Chao, H.C. Lin, W.H. Lin, C.C. Lo, An efficient flow control approach for SDN-based network threat detection and migration using support vector machine, in: IEEE Conference on e-Business Engineering, 2016, pp. 56–63..
[25] X.Y. Liu, J. Wu, Z.H. Zhou, Exploratory undersampling for class-imbalance learning, IEEE Trans. Syst. Man Cybern. Part B (Cybernetics) 39 (2) (2009) 539–550.
[26] A. Gupta, Machine learning with MATLAB (2016). https://au.mathworks.com/matlabcentral..
[27] S. Wang, S. Chandrasekharan, K. Gomez, S. Kandeepan, A. Al-Hourani, M.R. Asghar, G. Russello, P. Zanna, SECOD: SDN secure control and data plane algorithm for detecting and defending against DoS attacks, in: IEEE/IFIP Network Operations and Management Symposium, 2018..
[28] S. Wang, K.G. Chavez, S. Kandeepan, P. Zanna, The smallest software defined network testbed in the world: Performance and security, in: NOMS 2018–2018 IEEE/IFIP Network Operations and Management Symposium, IEEE, 2018, pp. 1–2..

[29] R. Caruana, A. Niculescu-Mizil, An empirical comparison of supervised learning algorithms, in: Proceedings of the 23rd international conference on Machine learning, 2006, pp. 161–168.
[30] J. Peng, P. Zhang, N. Riedel, Discriminant learning analysis, IEEE Trans. Syst. Man Cybern. Part B (Cybernetics) 38 (6) (2008) 1614–1625.
[31] R. Datti, B.B. Verma, Feature reduction for intrusion detection using linear discriminant analysis, Int. J. Eng. Sci. Technol. Citeseer (2010).
[32] B. Kamiński, M. Jakubczyk, P. Szufel, A framework for sensitivity analysis of decision trees, Central Eur. J. Oper. Res. 26 (1) (2018) 135–159.
[33] L. Yu, L. Wang, Y. Shao, L. Guo, B. Cui, GLM+: An efficient system for generalized linear models, in: IEEE International Conference on Big Data and Smart Computing (BigComp), 2018, pp. 293–300.
[34] B. Shin, J.H. Lee, T. Lee, H.S. Kim, Enhanced weighted k-nearest neighbor algorithm for indoor Wi-Fi positioning systems, in: 2012 8th International Conference on Computing Technology and Information Management (NCM and ICNIT), vol. 2, IEEE, 2012, pp. 574–577..
[35] J. Wu, S. Pan, X. Zhu, Z. Cai, P. Zhang, C. Zhang, Self-adaptive attribute weighting for naive bayes classification, Expert Syst. Appl. 42 (3) (2015) 1487–1502.
[36] J.G. Hsieh, Y.L. Lin, J.H. Jeng, Preliminary study on Wilcoxon learning machines, IEEE Trans. Neural Networks 19 (2) (2008) 201–211.
[37] C.W. Hsu, C.J. Lin, A comparison of methods for multiclass support vector machines, IEEE Trans. Neural Networks 13 (2) (2002) 415–425.
[38] R.E. Banfield, L.O. Hall, K.W. Bowyer, W.P. Kegelmeyer, A comparison of decision tree ensemble creation techniques, IEEE Trans. Pattern Anal. Mach. Intell. 29 (2007) 173–180.
[39] Open Networking Foundation, OpenFlow Switch Specification, ONF TS-024, vol 1.4.1..
[40] N. Jiang, Y. Deng, A. Nallanathan, J.A. Chambers, Reinforcement learning for real-time optimization in nb-iot networks, IEEE J. Sel. Areas Commun. 37 (6) (2019) 1424–1440.
[41] M.V. Mahoney, P.K. Chan, An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection, in: International Workshop on Recent Advances in Intrusion Detection, Springer, 2003, pp. 220–237..
[42] N. Ahuja, G. Singal, D. Mukhopadhyay, Ddos attack sdn dataset, Mendeley Data (2020), https://doi.org/10.17632/jxpfjc64kr.1.
[43] M.S. Elsayed, N.A. Le-Khac, A.D. Jurcut, Insdn: A novel sdn intrusion dataset, IEEE Access 8 (2020) 165263–165284.
[44] I. Ahmad, M. Basheri, M.J. Iqbal, A. Rahim, Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection, IEEE Access..