Journal of Information Systems and Telecommunication
Vol.10, No.2, April-June 2022, 120-131
http://jist.acecr.org
ISSN 2322-1437 / ESSN:2345-2773

Self-Organization Map (SOM) Algorithm for DDoS Attack Detection in Distributed Software Defined Network (D-SDN)

Mohsen Rafiee[1], Alireza Shirmarz[2*]

[1].Department of Computer Engineering, University of Mazandaran, Mazandaran, Iran
[2].Department of Computer & Electronic Engineering, Ale-Taha University, Tehran, Iran

Received: 05 Dec 2020/ Revised: 15 Sep 2021/ Accepted: 21 Nov 2021
DOI:

Abstract

The extend of the internet across the world has increased cyber-attacks and threats. One of the most significant threats includes denial-of-service (DoS) which causes the server or network not to be able to serve. This attack can be done by distributed nodes in the network as if the nodes collaborated. This attack is called distributed denial-of-service (DDoS). There is offered a novel architecture for the future networks to make them more agile, programmable and flexible. This architecture is called software defined network (SDN) that the main idea is data and control network flows separation. This architecture allows the network administrator to resist DDoS attacks in the centralized controller. The main issue is to detect DDoS flows in the controller. In this paper, the Self-Organizing Map (SOM) method and Learning Vector Quantization (LVQ) are used for DDoS attack detection in SDN with distributed architecture in the control layer. To evaluate the proposed model, we use a labelled data set to prove the proposed model that has improved the DDoS attack flow detection by 99.56%. This research can be used by the researchers working on SDN-based DDoS attack detection improvement.

Keywords: Software Defined Network (SDN); Distributed Controller; Distributed Denial-of-Service (DDoS); Self-Organizing Map (SOM); Learning Vector Quantization (LVQ).

1- Introduction

Internet extend has been raised across the world sharply, so internet technology usage rate among business and the social activities went up. The complexity of the traditional network architecture on the internet exposes the network specialist to a situation that makes the configuration and network control impossible, so the scientists proposed a new architecture called software defined network (SDN) to be used for future networks [1]. The SDN includes three layers, application, control, and data. There are various defined tasks for each layer and this structure makes the network much more programmable, flexible, and manageable [2][3]. The SDN, in addition to three layers, has three APIs (Northbound, Southbound, and East-West) to connect the layers and scale the controller with controllers' communication capability [2][4][5][6][7] which Fig.1 shows the layers and APIs, briefly. The data plane is composed of FEs which are simple forwarding elements. The control plane has the main role of decision-making in SDN. The controller can be a physical centralized or conceptual centralized controller. The conceptual centralized controller is composed of some controllers which are related together with east-west APIs. The data plane and control plane are connected by southbound API. The application layer is based on network applications and is connected with the control plane with northbound APIs.

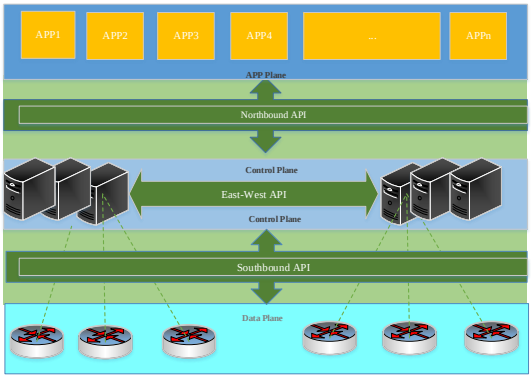

Fig. 1 SDN Architecture

✉ **Alireza Shirmarz**
a.shirmarz@aut.ac.ir

1-1- Problem Statement

One of the significant attacks on the internet is DDoS which has laid in the central attention of the published recent papers. The papers have proposed approaches to mitigate DDoS attacks with the use of machine learning (ML) or statistical methods. The DDoS attacks have been categorized in [8]. The DDoS is a malicious effort to disrupt the normal network flows that are done with fake traffic generating as if the service cannot be provided by the server or the network. The denial of service takes place because the computing resources are busy with fake traffic. According to Cisco's annual internet report which has been published in 2020, the number of DDoS attacks will double to 15.4 million by 2023 globally [9] as shown in Fig. 2.

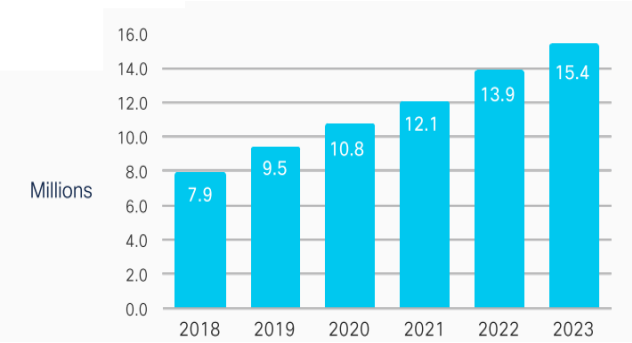

Fig. 2 The prediction of DDoS Attack growth by 2023 [9]

The limitation in memory and processing resources that exist in smart devices has caused the networks to become more susceptible to large-scale DDoS attacks. Internet of things (IoT) expands the use of smart devices which makes this issue more important [10]. For instance, the DDoS attacks which happened in well-known companies and organizations like CNN, Netflix, Twitter caused a denial of service in 2006 [11]. These reasons show the importance of DDoS attacks and their effects on IT. The taxonomy of the DDoS attacks' types are organized in a tree which is presented in Fig.3.

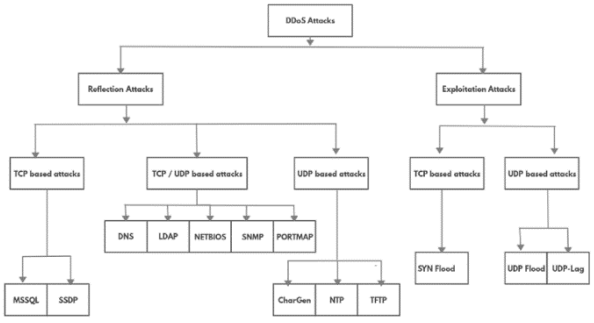

Fig. 3 DDoS Attacks Taxonomy [8]

One of the SDN traits is the (physical or conceptual) centeralized controller that has a global view of the network to make the optimized decision; therefore, it can make the network more secure from different attacks in comparison with the traditional networks. One of the SDN architecture drawbacks is a single point of failure of the controller; hence, the controller has been posed threats by the attackers that can make the controller denial of service. Due to this defect, distributed controllers architecture has been proposed which is addressed in this paper. This distributed controllers architecture has solved the single point of failure and mitigate the DDoS attack on the centeralized controller.

The first step to encounter DDoS is to detect the attack with a proper solution. Firewalling, intrusion detection system (IDS), and intrusion prevention system (IPS) can be developed and deployed in the application layer. The solutions in the application layer are facing problems like:

- Application layer workload
- Less smart behaviour
- Expert administrator
- Less Agility
- Interference in the defined rules
- The symmetry of the rules

1-2- Proposed Approach

In this paper, we propose the DDoS attack detection module in the controller with distributed architecture. The proposed solution is based on a machine learning method named self-organizing map (SOM) which is used for smart DDoS attack detection. There is a wide diversity in the flow patterns for DDoS attacks; therefore, it is needed to extract flows pattern automatically. The SOM is powerful in pattern extraction from real data. This pattern extraction lets us detect the new patterns for a DDoS attack. This algorithm can be used for classification and IDS as anomaly detection. The model is extracted from the dataset which is CICDDoS2019 and was obtained from the Canadian Institute for Cybersecurity, University of New Brunswick, Canada. This comprehensive dataset consists of 50063112 instances with 76 features along with 13 class labels to predict DDoS attacks [12]. The proposed approach will be compared with the related works and show that our model works with more accuracy. The proposed clustering algorithm labels the dataset and this method is compared with the labelled dataset. In this paper, a distributed architecture is used for the proposed model. The proposed model is simulated in Mininet and shows that the proposed model improves the DDoS attacks detection.

In this paper, the novelties that are proposed are:

- The Self-Organizing Map (SOM) method and Learning Vector Quantization (LVQ) are used for DDoS attack detection

- The DDoS attack detection model in distributed SDN controllers
- DDoS attack detection improvement in SDN

In the following section, the background of the concept will be discussed.

2- Background

2-1- SDN Functionality

In traditional networks, all switching and routing decisions are done in the switches, routers, firewalls and other equipment while these decisions are done in the centralized controller in SDN architecture. In SDN, there is only equipment instead of switches, routers, firewalls which is responsible for forwarding called forwarding element (FE). The other spec of SDN is the flow-based decision instead of the packet-based one. There is a definition for flow which is a sequence of packets with a common source and destination which is determined with five tuples (source IP, destination IP, source port number, destination port number and transport protocol). The packet header is extracted in each FE then the existence of the rule inside the flow table for the packet is examined. If the packet is the first packet of a flow and doesn't exist in the flow table, FE forwards the packet-in message including the packet header and payload to the controller. The controller makes a proper decision for each flow and exports the rule to each FE to fill the flow table. If the packet exists in the flow table, FE forwards the packet based on the action defined in the flow table. The controller decision for each flow has been made based on the network policy planned by the network administrators. The flow action is exported with the flow-mode message [13].

The rule of flow is embedded in the flow table which has been placed in FEs. The FE accomplish according to the action which is put in the flow table. The flow table is flushed in two cases, soft time-out, and hard time-out. The soft timeout depends on the idle flow entry which exists in the flow-table, and hard time-out refers to the period of the time in which the flow-table should be made empty by force. This mechanism makes memory space more efficient.

2-2- Self-Organizing Map Algorithm

The SOM algorithm has been proposed in 1982 by Kohonen [14]. This is an unsupervised learning algorithm that learns the patterns from complex datasets and clusters the data with noise. It is a neural network-based dimensionality reduction algorithm generally used to represent a high-dimensional dataset as two dimensional discretized pattern. Dimensionality reduction is performed while retaining the topology of data present in the original feature space. The clustering method is a k-means clustering performed on the mapping generated by SOM. As the first step, an artificial neural network is trained to generate a low-dimensional discretized representation of the data in the original feature space while preserving the topological properties; this is achieved through competitive learning. In SOM, the vectors that are close in the high-dimensional space also end up being mapped to SOM nodes that are close in low-dimensional space. K-means can be considered a simplified case of SOM, where the nodes (centroids) are independent of each other. K-means is highly sensitive to the initial positions of the centroids, and it is not suitable for a high-dimensional dataset. The two-stage procedure for clustering adopted in this study first uses SOM to produce the low-dimensional prototypes (abstractions) that are then clustered in the second stage using k-means. This two-step clustering method reduces the computational time and improves the efficiency of K-means clustering. Even with a relatively small number of samples, many clustering algorithms especially hierarchical ones become intractably heavy. Another benefit of the two-step clustering method is noise reduction. The prototypes constructed by SOM are local averages of the data; therefore, less sensitive to random variations than the original data. The weights of SOM were randomly initialized. During training, the weight vectors are updated based on the similarity between the weight vectors and input vectors which results in moving the SOM neurons/nodes closer to certain dense regions of the original data. The similarity between data points and SOM nodes during the weight update is evaluated based on Euclidean distance [15] as shown in Fig. 4. The main steps which are required in SOM are:

1) Train step: the neurons' network weights are determined with trained sets.
2) Map step: the winner neurons are chosen and clustered automatically.

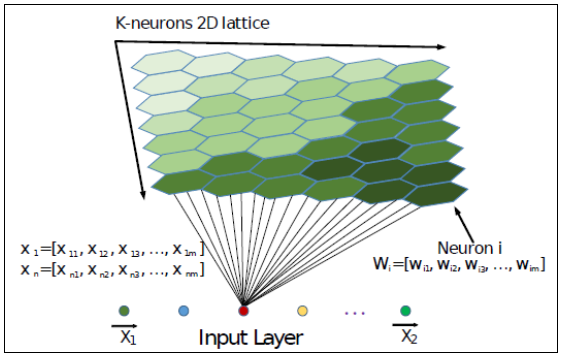

Fig. 4 The Self-organizing map [15]

The SOM algorithm defines some variables which are [16]:

- N: The number of training instances. These instances are shown in a set $[\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n]$
- S: The number of neurons has been mapped. These neurons are presented by the vector with m dimensions.
- The lattice radius is defined as follows:

$$R = \frac{\max(Map\ Width,\ MapHeight)}{2} \quad (1)$$

- λ is fixed and calculated as:

$$\lambda = \frac{N}{\log R} \quad (2)$$

- σ(t) is the winner neurons radius as shown in Fig. 5. This radius is calculated by the dependent period like t.

$$\sigma(t) = R \times \exp\left(-\frac{t^2}{\lambda}\right) \qquad t = 1, \ldots, n \quad (3)$$

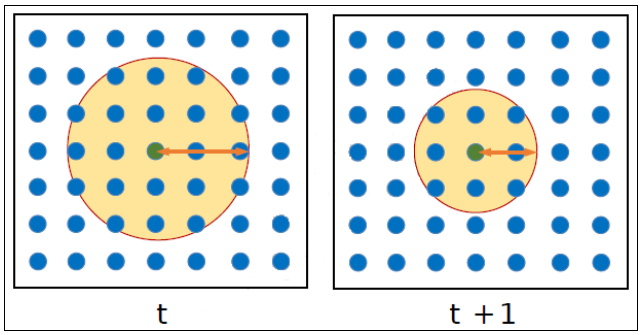

Fig. 5 The radius degradation in the map [14]

The SOM is computed in three phases:

1) **Competition Phase:** In this stage, neurons compete to choose the center of each cluster. For this purpose, the neurons are first given the initial value. The dimensions of these values are the same as the input data.

$$W_i = [\vec{W}_{i1}, \vec{W}_{i2}, \ldots, \vec{W}_{im}] \quad 1 \le i \le S \quad (4)$$

The cluster with the smallest distance to the input vector $x_k = [x_{k1}, x_{k2}, \ldots, x_{km}]$ wins the competition. There are several methods for determining the distance between neurons and the input vector. In this article, the Euclidean distance is used and can be expressed as follows:

$$D_{ist} = \sqrt{\sum_{j=0}^{m}(x_{kj} - W_{ij})^2} \quad (5)$$

2) **Collaboration Phase:** In this stage, the effect of winning i neurons on the learning of neighbouring neurons when applying input x is calculated. The greater the distance between neighbouring neurons and the winning neuron, the less effective it is. Experience has shown that it is best to consider a large neighbourhood first to include dead neurons. The magnitude of this effect is calculated from the following equation:

$$\Theta(t) = \exp\left(-\frac{D_{ist}^2}{2\sigma(t)^2}\right) \quad (6)$$

3) **Adaptation Phase:** In this stage, according to the equation under the weight of the neurons, they are optimized for the next repetitions so that over time, the whole map converges towards the input vector:

$$W(t+1) = W(t) + L(t) \times \Theta(t) \times (x_k(t) - W(t)) \quad (7)$$

The variable L(t) is the learning rate that decreases over time. It is calculated as:

$$L(t) = L_0 \times \exp\left(-\frac{t}{\lambda}\right) \quad (8)$$

Collaboration and adaptation are repeated to enter the mapping stage to complete network learning.

2-3- DDoS Attacks

To detect DDoS attacks, it is needed to know the characteristics of this attack. Generally, DDoS attackers disable the targeted victim by anomaly fake network traffic generating. There are different categories that have been proposed for DDoS attack types. In this paper, Figure 3 shows a classification of DDoS and includes two major general modes. The first DDoS attack model is based on the occupied victim system's bandwidth by large packets as if the system cannot service like the DNS service attack. The second model that can be done as the DDoS attack is to disrupt the main resources of the victim system such as memory and processor by sending unusual and abnormal packets such as syn-flood.

3- Related Works

In this section, we review SDN security researches related to DDoS attack detection mechanisms and address approaches similar to the proposed method. DDoS attack detection in SDN can be categorized into two methods which are based on statistical analysis methods and Machine Learning (ML) methods.

In [17]–[20] researches, statistical analysis is used to detect DDoS attacks in SDN. The entropy method is one of the most widely used statistical analysis methods to

detect DDoS attacks. Entropy is a parameter to measure randomness. That is, it determines the probability of an event occurring according to the total number of events. The higher the randomness, the higher the entropy. In [21] a threshold-based entropy method has been proposed to detect DDoS attacks in SDN. In a network, each host must receive new packets with a probability that is almost close to each other, in which case the entropy will be high. If one or more hosts receive too many packets, the randomness decreases and as a result, the entropy will decrease. According to this, a threshold being set for entropy, and the attack will be detected if the entropy value falls below this threshold. In [22] the controller periodically creates a hash table of destination addresses through the information received from the switches. The entropy of the destination addresses is then calculated by the phi-entropy method. A DDoS attack is detected if the entropy value is below the threshold for more than five consecutive windows.

The growth of Machine Learning knowledge makes many researchers use it in DDoS attack detection. One of the important and key steps in the use of Machine Learning is to select the appropriate features for learning the algorithm. In paper [23] It uses the Dynamic MLP method to select the optimal features. Polat et al It has used Support Vector Machine (SVM), Naive Bayes (NB), Artificial Neural Network (ANN), and K-Nearest Neighbors (KNN) machine learning methods to detect DDoS attack in SDN and has reached 98.3% in diagnosis accuracy with KNN method. Phan et al in [24] have proposed a Distributed SOM DSOM method to detect DDoS attacks in SDN. In this paper, the control plane architecture is Centralized with a POX controller. The proposed method also uses multiple self-organizing maps integrated with OpenFlow switches instead of a self-organizing one. Each DSOM in each switch processes the incoming traffic; hence, the processing load on the controllers will be divided between the switches. There is also a DSOM component in the application layer that is responsible for managing the performance of DSOM on switches. Due to the fact that in this method, the attack detection point is located in the switches and in the data layer, it is necessary to check all the packets passing through the switch by SOM, which consumes a lot of processing time and time. Braga et al in [25] introduce the Lightweight DDoS Flooding Attack method that detects a DDoS attack based on tracking suspicious input flows using self-organizing mapping. The control plane architecture is centralized with a NOX controller. The SOM features are extracted from the incoming traffic. Each instance then enters a SOM map to determine whether the incoming traffic flow is normal or malicious. This method includes three components Flow Collector, Feature Extractor and Classifier. The papers which are [26], [27] include other works that have used machine learning techniques to detect DDoS attacks in SDN.

T. Nam and et al have used SDN and proposed SOM and K-NN clustering to detect DDoS attacks in [28]. They have worked on DDoS attack detection and examined it with different k. They could find out the best accuracy for DDoS attacks with k=3. They could reach the accuracy %99.05 which is noticeable; therefore, we will compare our result with this paper.

The use of distributed controller architecture in the control layer and the application of distinctive features in the attack detection stage are the most important differences between our method and the above-mentioned works, which are discussed in this article.

4- The Proposed Scheme for DDoS Attack Detection in SDN

As mentioned in the proposed method, the components of DDoS attack detection are managed by controllers. The attack detection point will be at the level of the control layer of software-based networks. To achieve this goal, it is necessary to go through the four main steps shown in Fig.6 which in the proposed method will focus on the first and second steps:

1) Data Gathering: At this stage, appropriate statistical information should be obtained from network traffic so that normal traffic can be distinguished from attack traffic.
2) Attack Detection: A method should be implemented based on which the occurrence of the attack can be detected by entering the information collected in the previous step in the output. In this dissertation, machine learning in the proposed method is used.
3) Decision making: In this stage, it is determined what decisions should be made after identifying the attack. The main point of decision is in the network controller.
4) Execution: Converts the decisions made to the input of the flows and then applies them to switches and routers, such as deleting flow entry from the flow table.

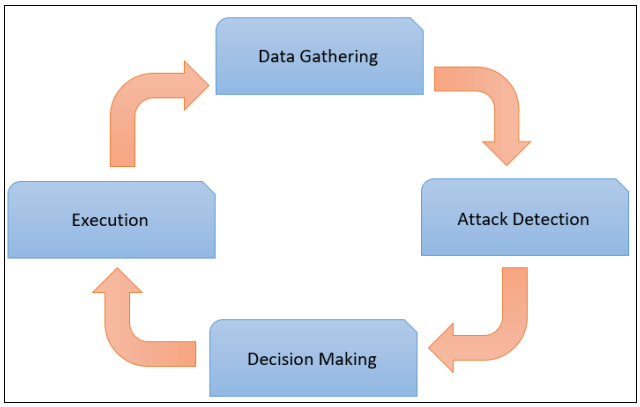

Fig. 6 Main four phases

In the following, first, the processing performed on the dataset used and how the self-organizing mapping algorithm works will be discussed. Then the attack simulation scenario is introduced along with the tools used to generate and analyze attack traffic.

4-1- Dataset Introduction

This paper uses the CICDDoS2019 dataset [8] compiled by the University of New Brunswick. This dataset includes normal traffic and the most up-to-date distributed denial-of-service attack traffic, which includes various types of attack traffic such as DNS, LDAP, SYN, etc.
To extract the feature, the CICFlowMeter two-way traffic flow generation tool was used. The output of this tool contains 76 features.

4-2- Data Pre-Processing

The Weka tool has been used to facilitate and expedite dataset processing. Weka is a Java-based machine learning tool developed at the University of Waikato in New Zealand. Weka allows users to extract useful information from the database. The heterogeneous dataset is used for classification. That is, the ratio of normal traffic to attack is very different, so-called unbalanced data. When the data is unbalanced, the performance of the detection algorithm cannot be properly evaluated. Because the neurons of the machine learning algorithm are biased towards the traffic class that has the largest number. Therefore, in order to balance the dataset, using the Weka tool, the Random Under Sampling method, which is one of the data Mining methods, has been used to eliminate the number of existing samples to reach balanced.
As mentioned, this dataset contains 76 features. One way is to use all of these features to detect an attack in a machine learning algorithm. However, due to a large number of features, it can prolong the processing time. On the other hand, the effect of all these features in identifying traffic related to a DDoS attack will not be the same. Some features will be more effective and some will be less effective. Choosing the most effective features is a major challenge in itself. As a result, the best solution is to be able to use all of the features in some way, as well as reduce the processing load by reducing the number of features. Therefore, in this paper, the feature extraction method [29] with the principle component analysis (PCA) has been used to obtain new features from the entire dataset. The method of calculating the new properties can be calculated as follows: where A_i is the main property of i, A'_i is the new property of i, V_{ij} is the component j of the vector i Eigenvector, n is the number of new properties obtained after PCA and m is the maximum number of main properties involved. In linear transmission are:

$$A'_i=V_{11}\times A_1+V_{12}\times A_2+\ldots+V_{im}\times A_m$$
$$=\sum_{j=1}^{m} v_{ij} \times A_j, \quad i=1,2,\ldots,n \quad (9)$$

To reduce the number of features, the PCA has been used and decreased the attributes from 76 to 23. Table 1 shows the obtained value after PCA processing that some are shown in the following table.

Table 1: PCA extracted features

#	PCA Extracted Features
1	-0.247Flow IAT Max - 0.246Fwd IAT Max - 0.244Idle Max - 0.241Idle Mean + …
2	-0.304Pkt Len Mean - 0.291Pkt Size Avg - 0.278Fwd Seg Size Avg + …
3	-0.235Subflow Bwd Byts - 0.235TotLen Bwd Pkts - 0.211Bwd Pkt Len Mean + …
4	-0.375Fwd Act Data Pkts - 0.353Tot Fwd Pkts - 0.353Subflow Fwd Pkts + …
5	0.476Active Mean + 0.441Active Max + 0.429Active Min + 0.306Flow IAT Min + …
6	-0.387Fwd PSH Flags - 0.387RST Flag Cnt - 0.294Flow Pkts/s - 0.293Fwd Pkts/s + …
7	-0.374Fwd PSH Flags - 0.374RST Flag Cnt - 0.333URG Flag Cnt + …
8	0.425CWE Flag Count + 0.405Down/Up Ratio - 0.372ACK Flag Cnt + …
9	-0.508Fwd IAT Min - 0.506Flow IAT Min - 0.229Flow IAT Mean + …
10	-0.464Bwd Pkt Len Min + 0.339CWE Flag Count + 0.292Init Fwd Win Byts + …
11	0.326Flow IAT Mean - 0.285Bwd IAT Max - 0.279Bwd IAT Tot + …
12	-0.715Fwd Header Len - 0.687Fwd Seg Size Min - 0.043Fwd Pkt Len Std + …

13	-0.418Init Bwd Win Byts + 0.342Bwd Pkts/s + 0.326Active Std + …
14	-0.464Active Std - 0.369Bwd IAT Mean + 0.31 Active Min + 0.304Idle Min + …
15	0.403Bwd IAT Mean + 0.318Bwd IAT Std - 0.294Fwd Pkt Len Std + …
16	0.687Bwd Header Len - 0.604SYN Flag Cnt - 0.246Bwd Pkts/s + …
17	-0.691SYN Flag Cnt - 0.658Bwd Header Len + 0.121CWE Flag Count + …
18	-0.641Bwd Pkts/s - 0.604Bwd IAT Min - 0.27Bwd Header Len + …
19	-0.643Bwd IAT Min + 0.558Bwd Pkts/s + 0.235Bwd Pkt Len Min + …
20	-0.38Init Bwd Win Byts - 0.377Bwd Pkt Len Min + 0.319Fwd Pkt Len Std + …
21	0.564Idle Std + 0.342Fwd Pkt Len Std - 0.282Active Std - 0.208Pkt Len Var + …
22	-0.368Pkt Len Var + 0.346Init Fwd Win Byts - 0.335Init Bwd Win Byts + …
23	0.66 Fwd Seg Size Min - 0.638Fwd Header Len - 0.157Idle Std + …

These acquired features have been rated based on the feature effectiveness. Therefore, these features that have been extracted based on PCA are sorted so that the feature, with more impact, has more value. This ranking was performed by the Filter method [29] Based on their evaluations. This ranking value is shown in Fig. 7.

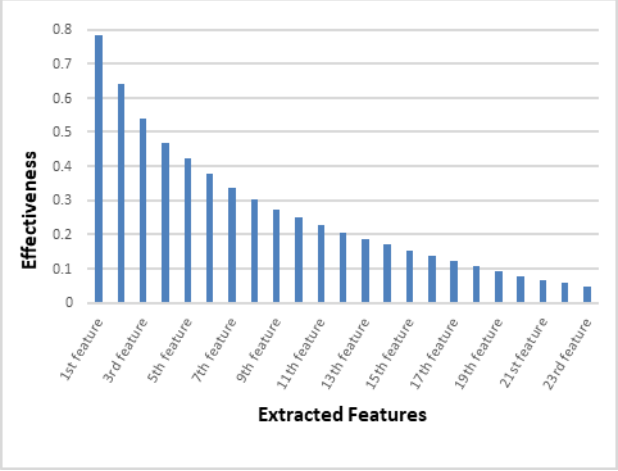

Fig. 7 Sorted extracted PCA features

The processed dataset is ready for training and evaluation in an attack detection system.

4-3- System Setup in SDN

Table 2 shows the specifications of the implementation environment. The Mininet emulator environment, which is licensed under the open-source BSD, is used to implement the Software Defined Network. Mininet provides a virtual environment in which all programs, switches, and code running on the actual system kernel, which can be a virtual machine, a cloud system, or a local system. The data layer uses the Open vSwitch (OVS). OVS is a multi-layer virtual switch that is Apache certified. These switches are programmable and support the OpenFlow protocol

Table 2: Environment Information

Name	Type / Name
OS	Ubuntu 18.04 64bit
CPU	Intel® Core™ i7-7700HQ CPU @ 2.80GHz × 3
RAM	8 GB
Simulator	Mininet
Switch	openvswitch 2.9.2
Controller	Floodlight Master
South API	OpenFlow1.3

In the control layer, the Floodlight Master controller is used in the network control section. Floodlight [30] is a Java-based open-source controller that supports both OpenFlow physical and virtual switches. This controller is Apache certified and has good scalability. The Floodlight Master version supports the architecture of distributed controllers efficiently. Fig. 8 shows the topology used in this scenario. One of these hosts is a Simple HTTP Server for web service that is considered as a victim of a DDoS attack. Of the other three hosts, one is considered as an attacker and the other as a normal user.

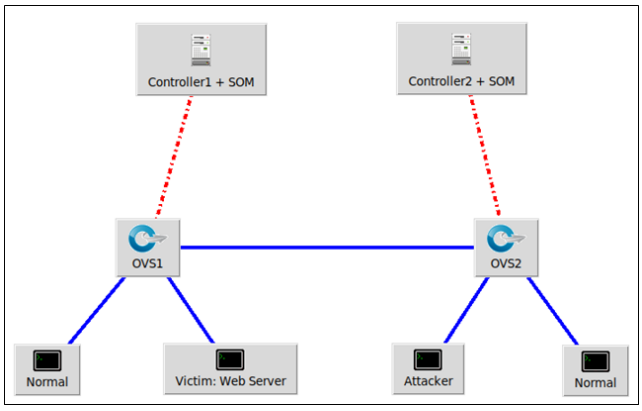

Fig. 8 Scenario Topology

Syn Flooding attack is used for the test. Scapy was used on the attacker host to implement this attack. To generate normal traffic, a shell script is used, which is sent to the webserver at the same time as the attack traffic.

In this architecture, the whole network can be divided into several independent domains in terms of geography and management. In our topology, the whole network is divided into two domains and each domain is managed by one controller. This architecture consists of two main parts:

1) Intra-domain communication, which includes the main function of the controller, i.e. sending policies and rules to the switches and receiving their status through the southbound interface.
2) Inter-domain communication, which includes the synchronization between the controllers through the east-west API as shown in Fig. 9.

Controller1:
[o.s.s.i.SyncManager] [1] Updating sync configuration ClusterConfig [allNodes={1=Node [hostname=127.0.0.1, port=6643, nodeId= domainId=1], 2=Node [hostname=127.0.0.1, port=6644, nodeId=2, domainId=2]}, authScheme=NO_AUTH]
[o.s.s.i.SyncManager] [1->2] Synchronizing local state to remote node

Controller2:
[o.s.s.i.SyncManager] [2] Updating sync configuration ClusterConfig [allNodes={1=Node [hostname=127.0.0.1, port=6643, nodeId=1, domainId=1], 2=Node [hostname=127.0.0.1, port=6644, nodeId=2, domainId=2]}, authScheme=NO_AUTH]
[o.s.s.i.SyncManager] [2->1] Synchronizing local state to remote node]

Fig. 9 Controller Synchronization

4-4- Flow Collection in Experiment

In the proposed method, the attack detection takes place in the control layer; That is, from the messages sent between the switch and the controller, abnormal flows related to the distributed denial-of-service attack must be identified. The challenge is that not all traffic packets pass through the control layer of the SDN network. To achieve this information, the flow entry in the flow table of each switch is used.

To solve the above challenge, the statistical information of network flows must be extracted from the messages exchanged in the OpenFlow protocol. The Floodlight controller consists of several components. One of these components is related to the collection of information from flow tables of network switches. This component sends a request to the switches at predetermined intervals, and the switches respond to the flow table information in response. Determining the amount of time interval, it takes to send a request is very important. If this time interval is considered too long, there will be a long delay in detecting the attack and if considered too short, the number of requests and responses between the controller and the switches will increase, increasing overhead. According to the paper [31], the time interval of the request is considered 3 seconds. Now, using this information, each controller will have its domain flow statistic.

4-5- Analysis of Detection Method

The self-organizing map consists of two layers, the input layer and the neural network layer. The number of neurons in the neural network layer indicates the number of output clusters that cluster the training data due to the unsupervised nature of this algorithm. This mode is useful when the training data is unlabeled and it is not clear to which category each input belongs. The data used in this article are labelled and fall into two general classes: Benign and DDoS. Therefore, the output of the detection system must be such that it can eventually map network neurons into these two classes. For this purpose, in addition to a self-organizing map, a supervised learning method called Learning Vector Quantization (LVQ) has been used. LVQ networks are a special type of competitive neural network that uses the idea of supervised learning and their main application is in classifying and recognizing patterns. This network is the development of a self-organizing map in a supervised state, and its learning method is quite similar to a self-organizing map, except that in LVQ only the winning neurons are moved and tuned each time, while in the self-organizing map, in addition to the winning neurons, the neighbouring neurons also move slightly. When this method is used in conjunction with a self-organizing map, it does the training twice, first clustering it unsupervised by the self-organizing map and then classifying it by learning vector Quantization. Fig. 10 shows the LVQ-SOM process. In this figure, from left to right, the first two layers are related to SOM mapping and the last two layers are related to LVQ.

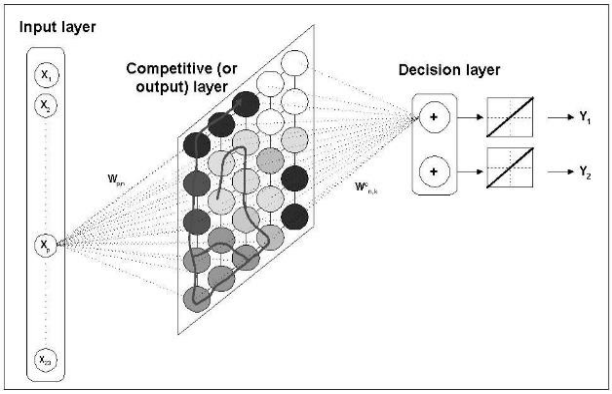

Fig. 10 LVQ-SOM [14]

The characteristics used in LVQ-SOM Learning are listed in Table 3.

Table 3: LVQ-SOM Parameters

Parameters	Value
Learning Rate	0.1
Epoch Limit	1440
Distance Type	Euclidean

5- Results and Analysis

To train the proposed artificial neural network is done with a dataset including different types of DDoS attacks as if it can discriminate varying DDoS types. The DDoS attacks which have been considered in this paper are SYN, DNS, LDAP, NTP and UDP attacks. Table 4 shows DDoS attack types distribution in this dataset. The number of attacked traffic and normal traffics are balanced in the training set. In this research, the DDoS attacks are not discriminated; hence, all DDoS attacks are combined as DDoS. To improve the training phase, the training set has been shuffled.

Table 4: DDoS Attacks for Training and Testing

Attack Type	# of Flows
SYN	30321
DNS	1985
LDAP	4007
NTP	13479
UDP	1119

In machine learning methods, evaluation metrics are divided into two stages; the Training phase and the test phase. In the training phase, evaluation metrics are used to optimize the algorithm. In other words, evaluation metrics are used to select the best solution to increase the estimation accuracy of the algorithm. While in the test phase, the evaluation metrics measure the efficiency of the model created in the classification of new data.

5-1- Evaluation Metrics

One of the most important evaluation metrics in two-tier classes is the Confusion Matrix. This matrix is 2 × 2, the rows of which represent the estimated categories and the columns of which represent the actual classes. In this matrix, four variables are defined, which are summarized in Table 5 as shown below.

Table 5: Evaluation variables

Evaluation variables	Definition
True Positive (TP)	Attack traffic identified as an attack
False Positive (FP)	Attack traffic detected as normal
True Negative (TN)	Normal traffic that is identified as normal
False Negative (FN)	Normal traffic identified as an attack

These variables can be used to derive other evaluation metrics:

- **Accuracy** shows the ratio of correct estimates to the total.

$$Acc = \frac{tp+tn}{tp+fp+tn+fn} \quad (10)$$

- **Error Rate** that shows the ratio of incorrect estimates to the total.

$$Err = \frac{fp+fn}{tp+fp+tn+fn} \quad (11)$$

- **Precision** indicates the ratio of positively estimated in positive class to total positive class. This criterion is used when the false positive rate is significant.

$$P = \frac{tp}{tp+fp} \quad (12)$$

- **Recall** is the ratio of positively estimated in positive class to the total number of samples estimated as positive. This criterion is used when the false negative is significant.

$$R = \frac{tp}{tp+fn} \quad (13)$$

- **F-Measure** is the harmonic mean of precision and recall.

$$FM = \frac{2 \times P \times R}{P+R} \quad (13)$$

- **Mean Square Error (MSE)** is the metric for evaluating the error of the training step

$$MSE = \frac{1}{n} \sum_{j=1}^{n} (P_j - A_j)^2 \quad (14)$$

The evaluation metrics are summarized in Table. 6. These metrics are used to compare our proposed model with the other similar approach.

Table 6: Evaluation metrics

Evaluation Metrics	Definition
Accuracy	The ratio of correct estimates to the total
Error rate (ER)	The ratio of incorrect estimates to the total
Precision	The ratio of positively estimated in positive class to the total positive class
Recall	The ratio of positively estimated in positive class to the total number of samples estimated as positive
F-Measure	Harmonic Mean of precision and recall

5-2- Performance Evaluation

One of the most important characteristics of neural networks is the number of neurons. Increasing the number of neurons does not necessarily increase the accuracy of the diagnosis. If the number of neurons is more than a certain limit, they will not improve the accuracy and even reduce the accuracy. Therefore, the number of neurons

must be selected in such a way as to maximize the accuracy of the neural network. There is no specific method or formula for determining the number of neurons and it can only be obtained experimentally.

The efficiency of the neural network used in the proposed method has been evaluated with several different neurons. It started with 100 neurons and continued until 2000 neurons. If the accuracy and error rate metrics are important, according to the diagrams in Fig. 11 and Fig. 12 of the case where the number of neurons is 1500, the accuracy of the neural network is 98.86%, which is the highest compared to other cases. Also, with this number of neurons, the error rate becomes 1.01%, which is the lowest error rate. We hope that our research can address some of the security challenges in SDN.

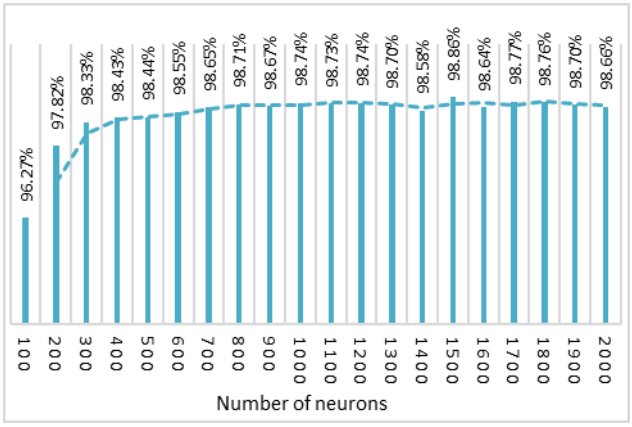

Fig. 11 Accuracy Rate for Different Number of Neurons

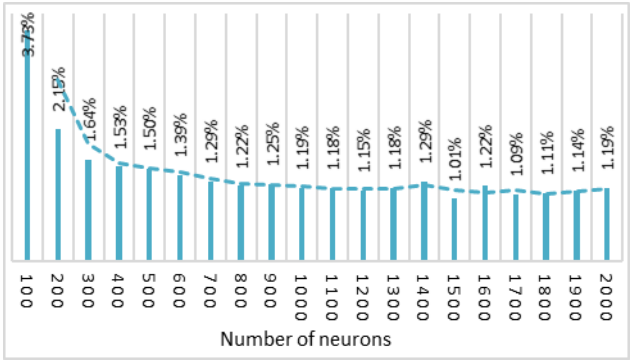

Fig. 12. Error Rate for Different Number of Neurons

If the metrics for precision and recall are important, according to the diagram in Fig. 13, which is drawn based on the F-measure, which is the equivalent of the two metrics for precision and recall, while the number of neurons is considered to be 1200 will have the most value. If the value of the precision metric is high, it means that the flows which are low likely to be attacked are recognized as normal flows. On the other hand, if the value of the recall metric is high, it means that a huge number of the normal flows are recognized as attack flows.

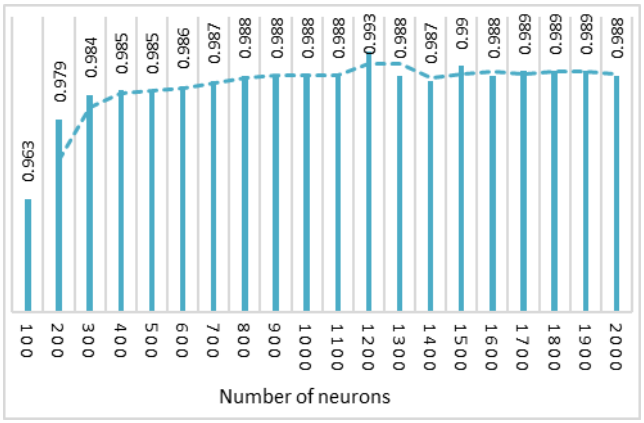

Fig. 13 F-Measure for Different Number of Neurons

Table 7 shows the evaluation metrics calculated in the proposed method for the number of different neurons.

Table 7: CICDDoS2019 Datasets Features

	Acc	Err	Precision	Recall	F-Measure	MSE
100	96.27%	3.73%	0.958	0.968	0.963	0.1683
200	97.82%	2.15%	0.973	0.984	0.979	0.1315
300	98.33%	1.64%	0.978	0.99	0.984	0.1134
400	98.43%	1.53%	0.98	0.989	0.985	0.1077
500	98.44%	1.5%	0.98	0.99	0.985	0.107
600	98.55%	1.39%	0.981	0.992	0.986	0.1025
700	98.65%	1.29%	0.984	0.99	0.987	0.099
800	98.71%	1.22%	0.985	0.991	0.988	0.0955
900	98.67%	1.25%	0.982	0.993	0.988	0.097
1000	98.37%	1.19%	0.985	0.997	0.988	0.0952
1100	98.73%	1.18%	0.986	0.991	0.988	0.0938
1200	98.74%	1.15%	0.984	0.993	0.993	0.0935
1300	98.7%	1.18%	0.985	0.991	0.988	0.0944
1400	95.58%	1.29%	0.983	0.991	0.987	0.0972
1500	98.86%	1.01%	0.988	0.991	0.99	0.0884
1600	98.64%	1.22%	0.984	0.992	0.988	0.0944
1700	98.77%	1.09%	0.985	0.993	0.989	0.0912
1800	98.76%	1.11%	0.986	0.992	0.989	0.0909
1900	98.7%	1.14%	0.984	0.993	0.989	0.0922
2000	98.66%	1.19%	0.983	0.993	0.988	0.0933

In order to better display the output of the work, we have made a comparison with one of the similar works. Nam et al. in [28] proposed two classification mechanisms to detect DDoS attacks in SDN with centralized controller architecture. These mechanisms are SOM + KNN and SOM with center-distributed classification and the features are Entropy of source IP, Entropy of source port, Entropy of destination port, Entropy of packet, protocol and the

Total number of packets. In this work, the monitor module collects the traffic information from the switches and after processing forward to the Algorithm module. The algorithm module classifies the network state as normal or under attack. If the network is under attack, then it generates an alert to the mitigation module. Then, the mitigation module generates the new policies and forwards these decisions to the switches as well as the server.

Our comparison was evaluated through the Detection Rate measurement (DR) and the False Alarm rate (FA), computed using Equations 15 and 16, respectively.

$$DR = \frac{tp}{tp + fn} \quad (15)$$

$$FA = \frac{fp}{tn + fp} \quad (16)$$

Table. 8 shows the results of our proposed model and SOM + KNN and SOM with center-distributed classification.

Table 8: Comparison results

Method	DR(%)	FA(%)
Proposed model	99.56	0.86
SOM + KNN	98.24	2.14
SOM with center-distributed	97.28	22.36

6- Conclusion

The main idea in this paper is a novel model to detect the DDoS attack, so a Self-Organizing Map (SOM) has been used to cluster the traffic datasets according to their similarity. The model has been designed and evaluated with the CICDDoS2019 dataset that had been labelled before. The simulation has been implemented in WEKA and the results show that the SOM works well with 1500 neurons. According to the labelled CICDDoS2019, the similarity is about 98.3% which is acceptable. To define DDoS traffic detection, a novel model has been proposed. The feature extraction has been done with the PCA method and trained with the CICDDoS2019 and LVQ. The proposed DDoS attack detection model has been developed in the controller that is the main layer of SDN architecture. The other momentous contribution is that the proposed model could protect the network from DDoS attacks with the distributed controllers' structure. The DDoS attack detection model has been implemented in the Floodlight controller in java language using the WEKA library. The simulation has been done with Mininet as an SDN emulation. The simulation results indicate that the proposed model could reach 99.56% accuracy to detect DDoS attacks while this model has been implemented in SDN architecture with distributed controllers. The proposed model has reached an acceptable accuracy, but the drawback of this model is the time consumption for clustering the traffics and limited dataset which can be the future work. The computation time can be considered as another future work.

References

[1] A. Shirmarz and A. Ghaffari, "An Autonomic Software Defined Network (SDN) Architecture With Performance Improvement Considering," J. Inf. Syst. Telecommun., vol. 8, no. 2, pp. 1–9, 2020.

[2] A. Shirmarz and A. Ghaffari, "Performance issues and solutions in SDN-based data center: a survey," J. Supercomput., 2020.

[3] A. Shirmarz and A. Ghaffari, "An adaptive greedy flow routing algorithm for performance improvement in a software- defined network," Int. Numer. Model. Electron. networks, Devices, Fields-Wiley online Libr., no. March, pp. 1–21, 2019.

[4] R. Masoudi and A. Ghaffari, "Software defined networks: A survey," J. Netw. Comput. Appl., vol. 67, pp. 1–25, 2016.

[5] Z. Zhao et al., "Autonomic communications in software-driven networks," IEEE J. Sel. Areas Commun., vol. 35, no. 11, pp. 2431–2445, 2017.

[6] A. Shirmarz and A. Ghaffari, "Taxonomy of controller placement problem (CPP) optimization in Software Defined Network (SDN): a survey," J. Ambient Intell. Humaniz. Comput., no. 0123456789, 2021.

[7] A. G. Alireza Shirmarz, "Automatic Software Defined Network (SDN) Performance Management Using TOPSIS Decision-Making Algorithm," J. Grid Comput., 2021.

[8] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," Proc. - Int. Carnahan Conf. Secur. Technol., vol. 2019-Octob, 2019.

[9] T. Cisco and A. Internet, "Cisco Annual Internet Report," 2020.

[10] "Legal Implications of DDoS Attacks and the Internet of Things (IoT)," 2016. [Online]. Available: https://www.dataprotectionreport.com/2016/12/legal-implications-of-ddos-attacks-and-the-internet-of-things-iot/.

[11] "Defending against Distributed Denial of Service (DDoS) attacks," 2020. [Online]. Available: https://www2.deloitte.com/ca/en/pages/risk/articles/DDoSattacks.html.

[12] "UNB Dataset." [Online]. Available: www . unb . ca / cic / datasets / ddos - 2019 . html.

[13] Q. Niyaz, W. Sun, and M. Alam, "Impact on SDN Powered Network Services Under Adversarial Attacks," Procedia - Procedia Comput. Sci., vol. 62, no. Scse, pp. 228–235, 2015.

[14] Teuvo Kohonen, The Basic SOM. 2001.

[15] T. V Phan, N. K. Bao, and M. Park, "Author ' s Accepted Manuscript Performance Bottleneck Handler for Large-sized Software- Defined Networks under Flooding Attacks Reference : Distributed-SOM : A Novel Performance Bottleneck Handler for Large-sized," J. Netw. Comput. Appl., 2017.

[16] Teuvo Kohonen, "The self-organizing map," in Proceedings of the IEEE, 1990, pp. 1464–1480.

[17] B. Yuan, D. Zou, S. Yu, H. Jin, W. Qiang, and J. Shen, "Defending against flow table overloading attack in software-defined networks," IEEE Trans. Serv. Comput., vol. 12, no. 2, pp. 231–246, 2019.

[18] M. Clayton, C. Batt, M. Clayton, and C. Batt, Communications and networking. 2019.
[19] M. Xuanyuan, V. Ramsurrun, and A. Seeam, "Detection and mitigation of DDoS attacks using conditional entropy in software-defined networking," Proc. 11th Int. Conf. Adv. Comput. ICoAC 2019, pp. 66–71, 2019.
[20] A. Ahalawat, S. S. Dash, A. Panda, and K. S. Babu, "Entropy Based DDoS Detection and Mitigation in OpenFlow Enabled SDN," Proc. - Int. Conf. Vis. Towar. Emerg. Trends Commun. Networking, ViTECoN 2019, pp. 1–5, 2019.
[21] S. M. Mousavi and M. St-hilaire, "Early Detection of DDoS Attacks against SDN Controllers," in International Conference on Computing, Networking and Communications, Communications and Information Security Symposiu, 2015, pp. 77–81.
[22] S. M. S. Mousavi and M. St-Hilaire, "Early Detection of DDoS Attacks in Software Defined Networks Controller," Thesis, pp. 77–81, 2014.
[23] M. Wang, Y. Lu, and J. Qin, "A dynamic MLP-based DDoS attack detection method using feature selection and feedback," Comput. Secur., vol. 88, p. 101645, 2020.
[24] T. V. Phan, N. K. Bao, and M. Park, "Distributed-SOM: A novel performance bottleneck handler for large-sized software-defined networks under flooding attacks," J. Netw. Comput. Appl., vol. 91, pp. 14–25, 2017.
[25] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in Proceedings - Conference on Local Computer Networks, LCN, 2010, pp. 408–415.
[26] A. Detection, S. Networking, and S. K. Dey, "Effects of Machine Learning Approach in Flow-Based," 2019.
[27] R. Santos, D. Souza, W. Santo, A. Ribeiro, and E. Moreno, "Machine learning algorithms to detect DDoS attacks in SDN," Concurr. Comput. , vol. 32, no. 16, pp. 1–14, 2020.
[28] T. M. Nam et al., "Self-organizing map-based approaches in DDoS flooding detection using SDN," Int. Conf. Inf. Netw., vol. 2018-Janua, pp. 249–254, 2018.
[29] T. Khalil, "A Survey of Feature Selection and Feature Extraction Techniques in Machine Learning," pp. 372–378, 2014.
[30] S. Rowshanrad, V. Abdi, and M. Keshtgari, "Performance evaluation of SDN controllers: Floodlight and Opendaylight," Int. Islam. Univ. Malaysia Eng. J., vol. 17, no. 2, pp. 47–57, 2016.
[31] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS Flooding Attack Detection Using NOX/ OpenFlow," in 35th Annual IEEE Conference on Local Computer Networks, 2010, no. January 2015