Self-Organizing Map-based Approaches in DDoS Flooding Detection Using SDN

Tran Manh Nam[a], Phan Hai Phong[a], Tran Dinh Khoa[a], Truong Thu Huong[a], Pham Ngoc Nam[a], Nguyen Huu Thanh[a]
[a]*Hanoi University of Science and Technology*, Hanoi, Vietnam

Luong Xuan Thang[b], Pham Anh Tuan[b], Le Quang Dung[b], Vu Duy Loi[c]
[b]*Ministry of Public Security,*
[c]*Vietnam Academy of Science and Technology*, Hanoi, Vietnam

***Abstract*—Distributed Denial of Service (DDoS) attack is one of the most long-lasting problems in network security. Recently, although the DDoS attack mechanisms are widely understood, the problems are becoming more frequent due to the similarity between DDoS attack and normal traffic. In this work, we propose two approaches of DDoS attack detection which are based on the Self-Organizing Map (SOM). The proposed algorithms with their detection architecture are implemented in the Software-Defined Networking (SDN) technology which has the flexibility and programmable abilities. The SDN controller allows us to quickly perform complex classification and detection algorithm. By deploying a testbed environment, we successfully evaluate our proposed algorithms in terms of both accuracy and computational overhead. The experimental results show that these algorithms can reduce the processing time while maintain the suitable accuracy rate.**

***Index Terms*—DDoS detection, SDN, lightweight, SOM, k-NN**

I. Introduction

Nowadays, the Internet is threatened by numerous types of cyber-attacks. Among these, Distributed Denial of Service (DDoS) attack is the most common and long-lasting problem. As reported by Verisign [1], the DDoS average attack peak size in the first quarter of 2017, is 14.1Gbps, has increased 26% compared to the fourth quarter of 2016. The biggest challenge of successful detection of DDoS attack is the similarity between the normal and attack traffic patterns. Besides, with the IoT revolution, the number of packets during the DDoS is very large and encumbers the network monitoring and analyzing process. Over half a million IoT devices located in several countries worldwide were compromised by Mirai botnet [2], so that massive DDoS Attacks powered by IoT bots continue to spread day by day. As a result, it strongly necessitates a powerful DDoS detection mechanism that can flexibly and easily apply to a network.

At this time, Software-Defined Networking (SDN) is a technology that has flexibility and programmable ability to quickly perform complex classification and detection algorithm on flow-based network traffic. Consequently, in this work, we propose a lightweight DDoS attack detection method that based on the SDN technology. This work mainly focuses on the optimization of DDoS classification approaches with following contributions:

- Firstly, we propose two techniques with different trade-off priorities to classify normal and DDoS attack traffic after being trained by Self-Organizing Map (SOM) [3]: (1) A $hybrid$ machine learning technique *SOM+k-NN* which has the accuracy of the k-Nearest Neighbor algorithm (k-NN) [4], [5] and uses the trained SOM to significantly speed up the classification stage, at which k-NN performs very poorly; and (2) a *SOM distributed-center* algorithm of classification, which is used after the training of SOM is complete. This heuristic technique gains the fast processing time while maintains the acceptable accuracy.
- Secondly, we propose the SDN-Based DDoS attack detection platform which is implemented in a testbed by using SDN switches. This testbed allows estimating the accuracy and processing time of several algorithms as well as making the comparison among them.

This article is organized as follows. In the second section we present the related work. Our SDN-based DDoS solution and algorithms are described in section III. Section IV shows the performance evaluation of our solution. Finally, Section V concludes the article and discusses future work.

II. Related Work

SDN technology is becoming a de facto standard for the future of networking [6] and provides many advantages as follows: (1) *centralized management and control*; (2) *flexible software-based network function*; and (3) *lightweight packet forward*. In the network security sector, these SDN's advantages allow developers to easily and flexibly update a classification mechanism for detecting abnormal attacks to the control plane of the network. The SDN-controller quickly collects information from the switches, analyzes and then sends back the operating decision to them. With this flexible and effective process, the SDN-based DDoS attack detection has attracted attention of research community recently.

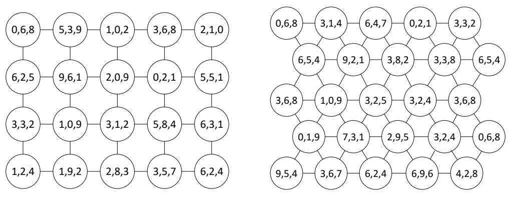
One of the well-known researches of SDN-based network security is *Lightweight* Flow-based Detection of DDoS [7] which periodically extracts six core features from the network traffic for training a Self-Organizing Map and detecting an attack. The SDN-controller collects these features from data planes and executes the algorithm on them. Although this method is lightweight, it does possess some following limitations. First, the SOM algorithm that was used in [7] is mainly used as visualization method for multidimensional data, not for traffic classification. The author did not clearly address the classifying method for a traffic attack detection, so that the integration of SOM with other traffic-classifying techniques is necessary. Secondly, some selected features in [7] such as number of packets per flow, bytes per flow and duration per flow itself cumulate over time. This information is actually representative of the connection between two hosts and not of the current instant. Therefore, it makes little sense to choose these as the indicator of a network at current time.

Besides, in [8], Avant-Guard is designed to migrate the bottleneck problem from saturation attacks. Avant-Guard provides a solution against only TCP SYN flood specifically by verifying a TCP 3-handshake process. Thereby it quickly detects TCP flood without collecting and analyzing data tuples. Similarly, Trung PV et al [9] proposed a solution in the SDN architecture as well which applies the Support Vector Machine method (SVM) and the so-called Idle Time-out Adjustment (IA) algorithm to detect DDoS attacks and classify data. The solution can deal with both TCP and ICMP flood that protects the network from resource exhaustion for the SDN controller and Openflow switch. Although [8] and [9] can detect DDoS attack with a high accuracy and little processing time, it only supports limited types of DDoS attacks.

From the above solutions, the DDoS detection system which is lightweight and supportable several types of DDoS attacks is necessary. From the perspective of a flexible and programmable network architecture such as SDN, it is a promising and suitable technology for contributing lightweight DDoS detection and mitigation system.

A. *Self-Organizing Map - SOM*

SOM is an unsupervised learning algorithm, a type of artificial neural network, that was first introduced by Kohonen [10]–[12]. Constructing SOM essentially means mapping from the high-dimensional space to usually two-dimensional space. The topological properties are preserved, which means nearby points in the original space are mapped to nearby points in the destination space. Hence, SOM is very useful in visualization of high-dimensional data.



(a) A rectangular SOM (b) A hexagonal SOM

Fig. 1: Two arrangement types of SOM

In essence, SOM is a map of topologically ordered neurons. The neurons can be positioned in a rectangular (Fig. 1a) or hexagonal grid (Fig. 1b). Each neuron consists of the number of weights which is equal to the dimension of input vectors to SOM. The training process is described as follows:

1) *Random initialization:* All neurons' weights are randomly assigned in the range from min to max of the respective feature in all training samples (uniformly distributed).
2) *Input sample:* The distances from a input sample, which is chosen, to all neurons are calculated and compared. The distance $d(\boldsymbol{p}, \boldsymbol{q})$ between two points **p** and **q** is calculated by using *Euclidean* equation (Eq.1).
$$d(\boldsymbol{p}, \boldsymbol{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \qquad (1)$$
3) *Find BMU:* The Best Matching Unit (BMU), $\boldsymbol{W}_b$, which is the nearest neuron to the input, is then determined:
$$\boldsymbol{W}_b = \operatorname*{argmin}_i d(\boldsymbol{V}, \boldsymbol{W}_i), i = 1, 2, ..., l \qquad (2)$$
$\boldsymbol{V}$ is the input vector, $\boldsymbol{W}_i$ is weight vector of i^{th} neuron in SOM, l is the number of neurons in the grid.
4) *Update neuron weights:* After the BMU is found, all neurons are updated with different rates. A neuron W_i is updated as follows:
$$\boldsymbol{W}_i(t+1) = \boldsymbol{W}_i(t) + \theta(b, i, t).\alpha(t).[\boldsymbol{V}(t) - \boldsymbol{W}_i(t)] \qquad (3)$$
where $\theta(b, i, t)$ is the neighborhood function and $\alpha(t)$ is the learning rate.
The nearer the BMU, the larger the neighborhood value is. Gaussian function is a usual choice for the neighborhood function given in Equation 4.
$$\theta(b, i, t) = e^{\frac{-d(\boldsymbol{W}_b, \boldsymbol{W}_i)^2}{2\lambda(t)}} \qquad (4)$$
Both $\lambda(t)$ and the learning rate $\alpha(t)$ are monotonically decreasing functions. Due to this, in the beginning, the self-organizing affects a broader region of the map and with a faster learning rate than in the end.
5) *Convergence:* Repeat the procedure *Input sample* to the *Update neuron weights* for a large number of cycles until some predefined convergence condition. After this training process, SOM will become a good approximation for the training input samples.

The Unified Distance Matrix (U-matrix) [13] is a representation of SOM showing the distance of a neuron to its neighboring neurons by grayscale. The nearer a neuron to its neighbors, the darker its color becomes. This is an efficient tool to visualize different clusters with SOM: different dark clusters are separated by light borders. Consequently, there have been many works aiming to apply SOM to support network anomaly detection by visualizing traffics [3], [7].

B. *k-Nearest Neighbors (k-NN)*

k-NN algorithm [4], [5] is one of the simplest among all machine learning algorithms that works well especially in low-dimensional feature space. k-NN is a non-parametric method which assumes nothing about the distribution of the data samples. k-NN is an instance-based learning which defers all generalization attempts until a query is made and performs the generalization by comparing new instances with training instances. Being such an algorithm, k-NN's computational cost of classifying new instances is very high due to the fact that all computational attempts take place in the classification phase.

The training instances for k-NN are feature vectors in a multidimensional feature space. Each vector has an explicit label. The training phase just consists of storing all vectors and their corresponding labels. k-NN subjects to the *curse of dimensionality* [14], [15] so normally, to avoid this, high-dimensional data should be preprocessed by dimensionality reduction [16], which is one of the reasons we choose SOM to preprocess data, as explained later.

In the classification phase, the Euclidean distances between the querying instance and all the training instances are measured. The measured Euclidean distances are then compared and k nearest instances are retrieved based on their distances with the querying instance. The majority class label among the chosen instances is the output class label for the querying instances. In case there are two types only (binary classification), k should be odd to avoid the equal-vote case.

Let C be the labels space:

$$\hat{f}(x_q) = \underset{c \in C}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(c, f(x_i)) \tag{5}$$

where $\delta(a, b) = 1$ if $a = b$ and $\delta(a, b) = 0$ otherwise.

III. SDN-based DDoS Detection Solution

A. System's Architecture

We propose a SDN-based DDoS detection and mitigation architecture that is described in Fig.2. The SDN controller communicates with OpenFlow-enabled switches and collects their information (flow tables). There are four additional modules in the SDN controller, namely *Monitor, Algorithm, Alert,* and *DDoS Mitigation*. The *Monitor* module collects information from OpenFlow-enabled switches, processes and forwards results to the *Algorithm* module. The *Algorithm* module determines the state of the network (normal or attack) and informs to the *Alert* and *DDoS Mitigation* modules if it is necessary. The *DDoS Mitigation* module contributes the policy and sends the forwarding decisions to related-switches in the network as well as servers. With this SDN-based solution, a classification algorithm with global network status is running by flexible up-to-date controller that stays on powerful dedicated server. So that our lightweight DDoS attack detection system can quickly perform complex algorithms.

The SDN controller periodically queries all its managed switches of their flow tables. Two consecutive flow tables will then be compared with each other to find out detailed information, representative features of all packets coming to the switch in that interval.

B. Representative Features

In abnormal behavior detection, since various extracted features have more randomness under attack state. In order to measure the degree of divergence, we use the concept of

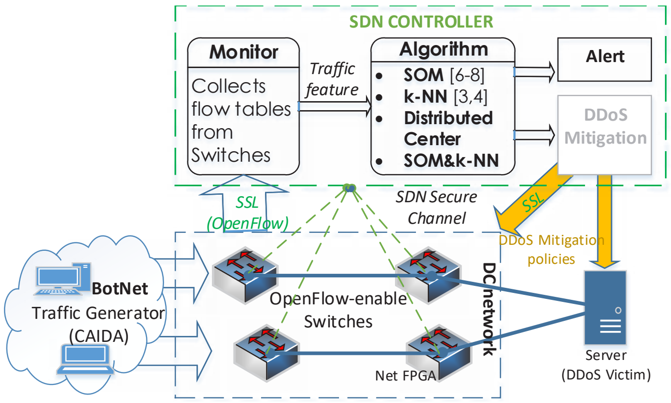


Fig. 2: Detection architecture

entropy [17] to build the representative features. If the information source has n possible values each with a probability of choice P_i, the entropy H is defined as follows:

$$H = -\sum_{i=1}^{n} P_i \log P_i \tag{6}$$

In order to detect several types of DDoS attacks, in this article we use five representative features analyzing and classification phrase. These classification's features are described as follows:

- *Entropy of source IP address (etpSrcIP):* During DDoS attack phase, the average number of source IP addresses would be very high, resulting in a high entropy.
- *Entropy of source port (etpSrcP):* Similar to IP spoofing, in a DDoS attack, TCP source port are usually also randomized and therefore manifest the same characteristics as IP source.
- *Entropy of destination port (etpDstP):* Similar to IP spoofing.
- *Entropy of packet protocol (etpProtocol):* In a SYN flood, UDP flood or ICMP flood attack, the majority of packets carries the same type of protocol, which is certainly different from normal traffic where different types of protocol are more reasonably distributed.
- *The total number of packets (totalPacket):* The number of packets during a given time period is one of the best indicator for DDoS attack. However, relying only on this feature alone can easily trick a system in case of a flash crowd.

Each of the above five features is a dimension of the weight vectors $\boldsymbol{W}$ of a neuron in SOM. Using these multi-dimensional vectors directly in raw form often causes the result to be biased to certain dimensions, as each dimension tends to have different units. Hence we use the *tanh-estimator* introduced by Hampel et al. [18] as a data preprocessing step to normalize the five-dimensional vectors to the range $[-1, 1]$, because of its efficiency and robustness compared to the commonly used *z-score* normalization [19]. In particular, a five-dimensional vector $(d_1, d_2, d_3, d_4, d_5)$ is normalized into $(n_1, n_2, n_3, n_4, n_5)$ by the following formula:

$$n_i = \frac{1}{2}\left\{\tanh\left(0.1\left(\frac{d_i - \mu_i}{\sigma_i}\right)\right) + 1\right\} \quad (7)$$

where $i = 1, 2, 3, 4, 5$; μ_i and σ_i are the mean and the standard deviation of the i^{th} dimension in the training data set, respectively. At the end of the normalization process, the calculated μ_i and σ_i for each dimension are saved for later use in real-time operation.

C. Algorithms

In this article, SOM is used as a preprocessing step before applying our proposed algorithms.

1) Self-Organizing Map training phase: After the normalization process using equation 7, the set of 5-dimensional vectors acting as the input of the training phase is obtained. Each vector is an instance, representing network state during a pre-determined interval. Those normalized instances are then ready to be fed to the system as training samples.

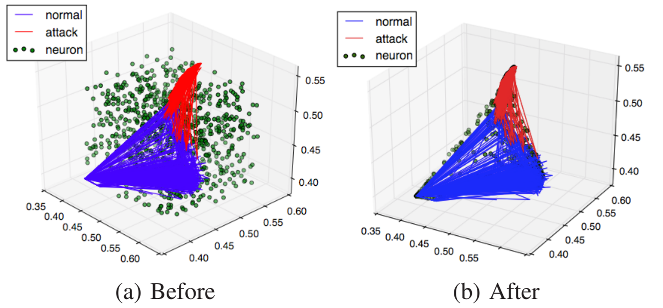


Fig. 3: SOM neurons in 3D before and after training (use 3 features to plot)

After training, all training instances are associated with their respective BMU. This step is important as it is the only source of labeling in SOM. Before training, neurons in SOM are randomly created with a uniform distribution, and therefore does not approximate the traffic pattern well enough compared to the state after training. This is illustrated in the region of associated sample instance in Fig. 4.

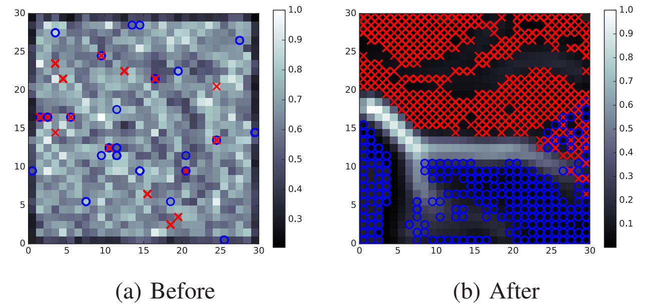


Fig. 4: The U-matrix and associating instances before and after training

2) Classification phase: After the preprocessing data by using SOM, two classification techniques are proposed in this article as follows:

2.1.) SOM + k-NN: we propose a combination algorithm, *SOM + k-NN*, for our DDoS attack detecting system. By using SOM as a layer between k-NN and the original data, we only need to work with a predefined number of samples (the number of neurons in SOM map) in a two-dimensional space. Here, instead of calculating the distance from the input sample to all neurons in SOM and then again calculate the distance to all training data correspond to the BMUs to find k nearest neighbor, Algorithm 1 skips the later step and find the most frequent label among all training data corresponding to k-BMUs. Hence, the computation complexity is reduced.

Algorithm 1 SOM +k-NN

```
1:  input: $\boldsymbol{W}_i = (w_i^1, w_i^2, ..., w_i^p)$, real-time input sample
        $\boldsymbol{V} = (v^1, v^2, ..., v^p)$
2:  // where $i = 1, 2, ..., l$
3:  // $l$ is the number of neurons in a $m \times n$ map of SOM
4:  $isAttack$ = False
5:  for $i = 1$ to $l$ do
6:      Compute distance $d(\boldsymbol{V}, \boldsymbol{W}_i)$
7:  end for
8:  Determine $k$-BMUs in the trained SOM
9:  Retrieve set $\boldsymbol{Q}$ of all training data corresponding to the
        determined $k$-BMUs
10: Determine $\hat{c}$ , the most frequent label (attack or normal)
        in $\boldsymbol{Q}$.
11: if $\hat{c} == 'attack'$ then
12:     $isAttack = True$
13: else
14:     $isAttack = False$
15: end if
16: output: $isAttack$
```

2.2.) SOM with center-distributed classification: Our second classification mechanism was inspired by the anomalous traffic detection technique presented in [3]. From now on, we will refer to the algorithm used in [3] as *SOM distributed-neurons*. In [3], for each input sample fed into the SOM, a corresponding winning neuron (BMU) must be located, which means distances between an input sample and the weight vectors of all neurons in SOM must be calculated. Furthermore, because [3] determines the input sample is anomaly or not based on its distance to the corresponding BMU, the flaw of this approach is that false negative cases tend to happen when the corresponding BMUs are found on the edge of the SOM.

To solve this, instead of finding BMUs for every input sample, we proposes a *SOM with center-distributed classification* mechanism which calculates the distances between each input sample to a universal reference point $\boldsymbol{G}$. In this proposed method, SOM is trained with only normal traffic samples. When the training of SOM is finished, a reference point $\boldsymbol{G} = (g_1, g_2, ..., g_n)$ is chosen. Each element of $\boldsymbol{G}$ is the median value of the respective feature among all the Self-Organizing Map neurons (Algorithm 2).

Then, as illustrated in Fig. 5, we choose a probability threshold σ, e.g. $\sigma = 0.95$, which indicates that 95% of the

Algorithm 2 SOM using Center-distributed classification

```
1: input: $\boldsymbol{W}_i = (w_i^1, w_i^2, ..., w_i^p)$, real-time input sample $\boldsymbol{V} = (v^1, v^2, ..., v^p)$
2: // where $i = 1, 2, ..., l$
3: // $l$ is the number of neurons in a $m \times n$ map of SOM
4: $isAttack$ = False
5: for $t = 1$ to $p$ do
6:     $Q^t = \{w_i^t : i = 1, 2, ..., l\}$
7:     $Q^t = sort(Q^t, order = increasing)$
8: end for
9: $\boldsymbol{G} = (g_1, g_2, ..., g_p)$
10: for $t = 1$ to $p$ do
11:     if $l$ is odd then
12:         $g_t = w^t_{(m \times n+1)/2}$
13:     else
14:         $g_t = \dfrac{w^t_{(m\times n)/2} + w^t_{(m\times n)/2+1}}{2}$
15:     end if
16: end for
17: Compute distance $d(\boldsymbol{V}, \boldsymbol{G})$
18: Find $F(x) = P(d' \geq x)$
19: Choose F(x) = $\sigma \implies$ find corresponding threshold $x = d'$
20: if $d(\boldsymbol{V}, \boldsymbol{G}) > d'$ then
21:     $isAttack = True$
22: else
23:     $isAttack = False$
24: end if
25: output: *isAttack*
```

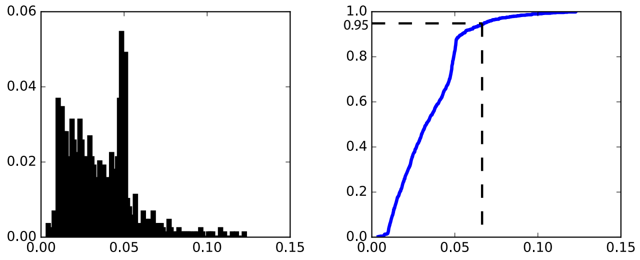
training data is within a hyper-sphere of radius d' centered around the reference point $\boldsymbol{G}$. The traffic corresponding to the input sample is flagged normal if its distance to $\boldsymbol{G}$ is less than the predefined distance threshold d' and flagged anomalous otherwise. There is a *false-positive/ false-negative trade-off* in choosing the heuristic value σ. The higher σ is, the more outliers it will capture, which means more false-positives. Likewise, the lower σ is, more false-negative will happen.

Moreover, in contrast to [3], in our second algorithm, only one distance needs to be calculated for each input sample. That is the distance between that input sample and the reference point $\boldsymbol{G}$. With our method, the processing time from the point when the input sample is fed into the SOM to the point when the classification decision is made is drastically reduced compared to [3].

IV. Performance evaluation

A. Testing Environment

For the testing experiments, many previous works [20]–[22] use the "DDoS Attack 2007" dataset [23] for the attack traffic data and one of the "CAIDA Anonymized Internet Traces Datasets" from 2008 to 2015 for the normal traffic data. However, due to the fact that these datasets are monitored in different locations, time and situation, they cannot represent a coherent traffic behavior. Therefore, in this experiment only the "DDoS Attack 2007" is used. This



(a) Histogram of the distribution of distances (b) Cumulative distribution of distances

Fig. 5: Distribution of distances from neurons to G

dataset contains approximately one hour of traffic before and during a DDoS attack on August 4, 2007, in which the first half an hour is normal traffic and the rest is attack traffic. We use POX [24] as our SDN controller due to the ease and efficiency in implementing the detection algorithms in Python. POX is run on a system of Intel's Core i5 processor and 4GB of RAM. We use an OpenFlow-enabled NetFPGA 1G as our switch [25]. For traffic generator, *TcpReplay* is used to regenerate the traffic trace from the "DDoS Attack 2007" dataset to the server computer through the switches' network. The *Monitor* module of SDN controller periodically collects all the statistic from the OpenFlow-enabled switches. According to our experiments, collecting these statistics once every 5 seconds provides reasonable trade-off between the accuracy and the performance of the network.

One of the key information here is which hyper-parameters are chosen for the initialization and training of SOM. The map is chosen in this work to be a 30 x 30 map. This allows an acceptable resolution for the map to correctly approximate the training data and also reasonable enough for the computational stress. Both the learning rate and the topological neighborhood radius are decreased with time. Gaussian function is used as topological neighborhood function. We use a Python implementation of SOM [26] to train the map, which is then embedded into the SDN Controller.

Two-thirds of the initial dataset is used for the training phase and the remaining data is used for the classification phase. In the classification phase, four algorithms are tested: k-NN, SOM + k-NN, SOM distributed-neurons and SOM distributed-center. The algorithms are tested with different hyper-parameters to measure their respective accuracy and processing time.

B. Experimental results

k = 3 is chosen in both k-NN and SOM + k-NN. As can be seen in Fig. 6, when k varies from 1 to 80, the respective processing time increases from about 2.7 ms to 3.2 ms, a 18.5% increase, meanwhile, the detection accuracy keeps nearly constant. Therefore, it makes little sense to choose much larger k. Letting $k = 1$ is not a reasonable choice since there can be times the BMU does not match with any pre-trained input sample, in which case the in effect k is

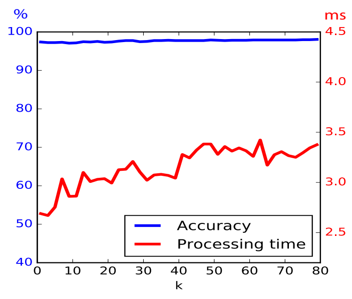


Fig. 6: Variation of SOM + k-NN algorithm

larger than 1, be it 2 or 3. Hence, $k = 3$ is a reasonable choice.

TABLE I: Comparison of 4 algorithms

Algorithm	DR(%)	FPR(%)	Processing Time(ms)
k-NN (k=3)	99.05	2.74	23.933
SOM + k-NN (k=3)	98.24	2.14	2.810
SOM distributed-neurons(σ=0.95) [3]	88.23	5.08	1.641
SOM distributed-center (σ=0.95)	97.28	22.36	0.004

Table I shows the Detection Rate (DR) and False Positive Rate (FPR) of referenced algorithms and our two proposed solutions, *SOM + k-NN* and *SOM distributed-center*. As we can see in the table, k-NN has the best accuracy but also possesses the largest processing time. Compared to k-NN, when integrated with SOM, the processing time is much better but both the detection rate and false positive rate needs to pay a price. The processing time of k-NN depends on the number of training samples and normally this number is much larger than the number of neurons in SOM, which explains for the results.

Compared to other algorithms, the SOM distributed-center algorithm has a much lower processing time, but also a much more tolerance for negative-bias, reflected in the detection rate and the false positive rate. It has the fastest processing time due to the fact that no sorting operation (which can contains up to thousands of elements or more) are needed.

V. Conclusion

In this paper, we have presented a DDoS detection system using SDN as well as two classification algorithms using Self-Organizing Map to classify the current network state as normal or attack. By using five representative features and deploying on SDN technology, the system is lightweight and can work well with several types of DDoS attacks. The proposed algorithms have proved to have better performance, given different priorities, than traditional detection algorithms. In the current work, all the features are selected manually such as to best represent the network traffic. However, these features may be different for each type of host that needs to be protected and for each type of attacks that can happen. As part of our future work, methods that attempt to automate the choosing of a feature are planned to be investigated. Furthermore, both our two current classification algorithms treat each instant of traffic separately, thus completely removing the chronological connections between them. Algorithms that can take advantage of this information would certainly be able to replace current classification algorithms.

References

[1] VeriSign DDoS Attack Trends - Q1, 2017. http://www.verisign.com/.

[2] Mirai IoT Botnet Description and DDoS Attack Mitigation. https://www.arbornetworks.com/blog/asert/mirai-iot-botnet-description-ddos-attack-mitigation/.

[3] Manikantan Ramadas, Shawn Ostermann, and Brett Tjaden. Detecting anomalous network traffic with self-organizing maps. In *Recent Advances in Intrusion Detection*, pages 36–54. Springer, 2003.

[4] Thomas M C. and Peter E H. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 1967.

[5] Richard O Duda, Peter E Hart, et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.

[6] Ramos F. M. V. Verissimo P. E. Rothenberg C. E. Azodolmolky S. Uhlig S. Kreutz, D. Software-Defined Networking : A Comprehensive Survey. *Proceedings of the IEEE*, 103(1), 2015.

[7] Rodrigo B., Edjard M., and Alexandre P. Lightweight DDoS flooding attack detection using NOX/OpenFlow. In *Local Computer Networks, 2010 IEEE 35th Conference on*, pages 408–415. IEEE, 2010.

[8] P.Porras S.Shin, V.Yegneswaran and G.Gu. Avant-guard: scalable and vigilant switch flow managament in software-defined networks. In *SIGSAC Conf. Com. Commun. Sec.*, pages 413–424. ACM, 2013.

[9] Trung P.V., Toan T.V., Tuyen D.V., Huong T.T., and Thanh N.H. Openflowsia: An optimized protection scheme for software-defined networks from flooding attacks. In *IEEE Sixth ICCE,*. IEEE, 2016.

[10] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982.

[11] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[12] Teuvo Kohonen and Panu Somervuo. Self-organizing maps of symbol strings. *Neurocomputing*, 21(1):19–30, 1998.

[13] Alfred Ultsch. *U*-matrix: a tool to visualize clusters in high dimensional data*. Fachbereich Mathematik und Informatik Marburg, 2003.

[14] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When Is "Nearest Neighbor" Meaningful? In *Database theory—ICDT'99*, pages 217–235. Springer, 1999.

[15] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *The thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

[16] Imola K Fodor. *A survey of dimension reduction techniques*. UCRL-ID-148494, Lawrence Livermore National Laboratory, 2002.

[17] Claude E S. A mathematical theory of communication. *SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[18] Frank R Hampel, Elvezio M Ronchetti, Peter J Rousseeuw, and Werner A Stahel. *Robust statistics: the approach based on influence functions*, volume 114. John Wiley & Sons, 2011.

[19] Anil J., Karthik N., and Arun R. Score normalization in multimodal biometric systems. *Pattern recognition*, 38(12):2270–2285, 2005.

[20] Wesam Bhaya and Mehdi Ebady Manaa. A Proactive DDoS Attack Detection Approach Using Data Mining Cluster Analysis. *Journal of Next Generation Information Technology*, 5(4):36, 2014.

[21] Mehdi Barati, Ammar Abdullah, Nur Izura Udzir, and el al. Distributed Denial of Service detection using hybrid machine learning technique. In *Biometrics and Security Technologies (ISBAST), 2014 International Symposium on*, pages 268–273. IEEE, 2014.

[22] V Aghaei-Foroushani and AN Zincir-Heywood. Tdfa: Tracebackbased defense against ddos flooding attacks. In *The 28th IEEE International Conference on Advanced Information Networking and Applications (AINA), Victoria, Canada, page*, 2014.

[23] The CAIDA UCSD "DDoS Attack 2007" Dataset. http://www.caida.org/data/passive/ddos-20070804_dataset.xml.

[24] The POX Controller. https://github.com/noxrepo/pox.

[25] OpenFlow NetFPGA project. https://github.com/NetFPGA/netfpga/wiki/OpenFlowNetFPGA100.

[26] The MiniSom project. https://github.com/JustGlowing/minisom.