Novaes, MP.; Carvalho, LF.; Lloret, J.; Lemes Proença, MJ. (2021). Adversarial Deep Learning approach detection and defense against DDoS attacks in SDN environments. Future Generation Computer Systems. 125:156-167. https://doi.org/10.1016/j.future.2021.06.047

Adversarial Deep Learning approach detection and defense against DDoS attacks in SDN environments

Matheus P. Novaes[a], Luiz F. Carvalho[b], Jaime Lloret[c,*], Mario Lemes Proença Jr.[d]

[a]*Electric Engineering Department, State University of Londrina (UEL), Londrina, Paraná, Brazil*
[b]*Computer Engineering Department, Federal Technology University of Paraná (UTFPR), Apucarana, Paraná, Brazil*
[c]*Integrated Management Coastal Research Institute, Universitat Politecnica de Valencia, Valencia, Spain*
[d]*Computer Science Department, State University of Londrina (UEL), Londrina, Paraná, Brazil*

Abstract

Over the last few years, Software Defined Networking (SDN) paradigm has become an emerging architecture to design future networks and to meet new application demands. SDN provides resources for improving network control and management by separating control and data plane, and the logical control is centralized in a controller. However, the centralized control logic can be an ideal target for malicious attacks, mainly Distributed Denial of Service (DDoS) attacks. Recently, Deep Learning has become a powerful technique applied in cybersecurity, and many Network Intrusion Detection (NIDS) have been proposed in recent researches. Some studies have indicated that deep neural networks are sensitive in detecting adversarial attacks. Adversarial attacks are instances with certain perturbations that cause deep neural networks to misclassify. In this paper, we proposed a detection and defense system based on Adversarial training in SDN , which uses Generative Adversarial Network (GAN) framework for detecting DDoS attacks and applies adversarial training to make the system less sensitive to adversarial attacks. The proposed system includes well-defined modules that enable continuous traffic monitoring using IP flow analysis, enabling the anomaly detection system to act in near-real-time. We conducted the experiments on two distinct scenarios, with emulated data and the public dataset CICDDoS 2019. Experimental results demonstrated that the system efficiently detected up-to-date common types of DDoS attacks compared to other approaches.

Keywords: Adversarial Attacks, DDoS, Deep Learning, GAN, SDN

1. Introduction

Nowadays, computer network systems have become complex structures for management and control. The main reason is the number of heterogeneous devices that make up the network. The Software Defined Networking paradigm (SDN) introduced tools to simplify configuration and management, in addition to enabling more significant innovation in communication networks. The SDN paradigm enables centralized network management. Where the network control is dissociated from the data forwarding plane [1, 2]. In the SDN architecture, the control plane is enhanced by centralized network control. The centralization of the control plane provides a global view of the network topology. Moreover, it enables the network traffic flow manipulation on runtime through an open and well-defined software interface [3, 4]. However, the centralized control plane can be a point of vulnerability, mainly targets of Distributed Denial of Service (DDoS) attacks [5, 6, 7, 8], which is easily flooded by many malicious requisitions. As a result, the controller may become unavailable to process normal user requisitions.

Security mechanisms are applied for detecting and preventing network systems from the actions of malicious agents. For this purpose, Network Intrusion Detection System (NIDS) is a widely used technique against Internet-based attacks [9, 10]. NIDS provides a set of tools able to recognize abnormal network behaviors automatically. A NIDS can be implemented as signature-based, anomaly-based, or hybrid. In signature-based NIDS, a database of known attack pattern signatures is used to recognize intrusions. This means, an intrusion is detected when there is a match between the stored patterns and the current behavior of network activity. Anomaly-based approach focuses in generating a normal behavior profile-based historical network data. When the predicted behavior and the current behavior diverge from one another, an anomaly is detected. The main advantage of this approach is the detection of zero-day and unknown attacks. Consequently, different techniques have been used to detect anomalies [11]; recently, NIDS has been proposed by applying deep learning techniques [12, 13].

Deep learning (DL) is a branch of Machine Learning (ML), where its application has become a trend, mainly due to the abstraction and generalization capacity in the learning process in many domains.[14, 15]. DL algorithms provide deep architectures formed by multiple layers of processing units to extract high-level abstraction from raw data. In literature, DL models such Convolution Neural Network (CNN) [16], Deep Boltzmann Machine (DBM) [17], Long Short-Term Memory (LSTM)[18], Recurrent Neural Network (RNN) [19], and Stacked

*Corresponding author
Email address: jlloret@dcom.upv.es (Jaime Lloret)

Autoencoder (SAE) [20] have been applied in many areas [21, 22, 23, 24], it includes audio processing [25], autonomous systems [26], cybersecurity [13], image and video recognition [27], and natural language processing [28]. In these fields, DL algorithms have shown their outperformance over classical ML algorithms related to classification tasks, dimensionality reduction, and feature learning algorithms.

Regarding security and management issues, deep learning technology can easily extract and learn characteristics and patterns from the network's behavior, providing useful insights for attack detection. Besides, with the fast increase in the number of users, network traffic becomes complex, and deep neural networks are powerful in reducing the network traffic complexity analysis due to their ability to learn massively complex data representations, and to handle it without human efforts [29].

Despite the extensive applications of DL algorithms in NIDS, recent studies have claimed that deep neural networks are sensitive to adversarial examples [30, 31, 32], which are performed by malicious agents to mislead DL algorithms in detecting attacks. Adversarial examples are instances with feature perturbations that are intended to cause a deep neural network to misclassify it. According to X. Zhang *et al.* [33] adversarial training can be applied to protect the system against adversarial example threats. In practical terms, this means adding adversarial data examples into the original dataset and using it in the model training phase. Generative Adversarial Network (GAN) framework enables to generate adversarial examples by adversarial training [34]. This GAN property can improve the NIDS performance in detecting adversarial attacks. By training the generator and the discriminator simultaneously in an adversarial way, the discriminator improves the detection rates using the generated adversarial examples and updates itself against them.

Given the significance of this issue, there is a need to ensure the security of network systems. The use of tools to support management and security activities is essential. Also, these tools must operate in an automated manner to facilitate identifying the occurrence of anomalous events and take countermeasures to minimize the effects caused by malicious agents. In this way, we present a novel anomaly detection system based on adversarial training by applying Generative Adversarial Network (GAN) for detecting and defending against up-to-date DDoS attacks.

The main contributions of this paper are listed as it follows:

- This work proposes a detection and defense system against adversarial DDoS attacks through an Adversarial Deep Learning approach, which provides a more accurate detection rate and less sensitive to adversarial examples;
- We conduct the experiments on two scenarios through up-to-date common DDoS attacks, such as NTP, DNS, LDAP, MSSQL, NetBIOS, SNMP, SSDP, UDP, UDP-Lag, WebDDoS (ARME), SYN e TFTP;
- The proposed system collects and analyzes the network traffic every second, enabling the anomaly detection system to act in near-real-time, that means, response time in up to 1 second;
- Compares the efficiency of the proposed system with different deep learning methods present in literature for detecting DDoS in SDN.

The remainder of this paper is organized as follows: Section 2 presents the related works; Section 3 introduces concepts and the structure of the proposed system; Section 4 presents the experimental scenarios and discusses the results achieved; Finally, Section 5 presents the conclusion and the future work.

2. Related Work

Li *et al.* [35] proposed an adversarial-example attack method to mislead the in-cloud firewall-equipped Android. The authors applied a variant of GANs, called bi-objective GAN, to generate adversarial examples. The bi-objective GAN has two Discriminators. The first one attempts to distinguish malicious examples from benign examples, and the second tries to distinguish adversarial examples from normal examples. In the experiments, it was defined benign apps come from Tecncet Myapp and AndroZoo, and the malicious apps are from VirusShare. According to experimental results, over 95% of the adversarial examples generated were detected as benign by the firewall.

Zhang *et al.* [33] applied a Monte Carlo tree search algorithm (MCTS) to generate adversarial examples of cross-site scripting (XSS) attacks. The authors also used a GAN framework to improve the intrusion detection model to protect against adversarial attacks. In the experimental phase, to generate new types of XSS attacks, the CICIDS 2017 is used. It was extracted XSS attack traffic examples and normal traffic examples from the dataset. The GAN detection model reached a precision rate above 99% to detect XSS attacks and its adversarial examples.

Grosse *et al.* [36] used the adversarial training technique for malware detection against adversarial examples. First, a Deep Neural Network (DNN) model was trained on the original dataset, composed of benign and malicious applications. After that, it was applied crafted adversarial examples. With those new samples, the DDN model was retrained to improve the model's generalization, and also it makes the DNN less sensitive to adversarial examples. The dataset used to evaluate the proposed model was the DREBIN dataset.

Distributed Denial of Service is the most common attack performed against cloud computing environments [37, 38]. Some solutions have been developed to address this challenge [39, 40]. Velliangiri and Pandey [41] proposed an approach that combines fuzzy and taylor-elephant herd optimization (FT-EHO) inspired by the Deep Belief Network (DBN) for detecting DDoS attacks. The proposed approach taylor-elephant herd optimization has three modules: feature extraction, feature selection, and classification. The first module extracts features from raw packets, and the feature selection module selects the best features by applying the Holoentropy method. Finally, the classification module detects the DDoS attacks using FT-EHO. Kushwah and Ranga [42] also proposed a system to detecting DDoS attacks in cloud computing environments. The proposed system is based on a voting extreme learning machine (V-ELM),

in which the majority results of artificial neural networks' outputs is used to get the final decision. According to the experiments' results, several ELMs can increase detection accuracy and reduce false alarms.

Akcay *et al.* [43] introduced a novel anomaly detection approach for images, using a conditional generative adversarial network. The proposed approach can learn the generation of high-dimensional image space and the inference of latent space. In the generator was applied an encoder-decoder-encoder to map the input image to a lower dimension vector. The generated image is mapped to its latent representation by applying an additional encoder network. The proposed approach learns the normal samples' data distribution, minimizing the distance between generated images and the latent vectors. An anomaly behavior is detected if a sample deviates a threshold from the distribution learned.

As mentioned before, Deep Learning algorithms have shown enhanced results for detecting normal DDoS attacks. However, few models are designed for detecting adversarial examples of DDoS attacks. The adversarial DDoS attacks refer to the type of violation to carry out a network attack causing misclassifications, where an attack is classified as normal (false negative). Undetected DDoS attacks can inflict damage on network resource availability [44]. For this purpose, we designed a system that applies adversarial training to overcome this vulnerability and improve the anomaly detection event rates in SDN.

3. Proposed System

According to previous studies [45, 46], the DDoS attacks are commonly carried out by malicious agents against network systems. Centralization of network control on a controller in the SDN architecture becomes a vulnerability exploited by DDoS attacks. In this regard, we proposed a system that acts in the application plane to detect and mitigate this threat. In Figure 1, a flowchart depicts the proposed system architecture. The system can be summarized in the following steps:

1. Data Collection: Every second, the controller collects IP flows from the tables flows of the switches that belong to the network.
2. Data Processing: In this step, categorical IP flows features (e.g., destination/ source IP addresses and ports numbers) are casting in numerical features.
3. Adversarial Deep Learning Anomaly Detection: The module analyzes the network's behavior and detects the occurrence of DDoS attacks.
4. Mitigation: In case a DDoS is detected, the mitigation module takes countermeasure to minimize the damages.

3.1. Data Collection

Data collection is a fundamental step for continuous monitoring of network activities. Besides, it is a prerequisite for detecting possible anomalies. For this purpose, the proposed system contains a module for acquisition of traffic data. In that module, the network information is collected from the switches using the OpenFlow protocol. The protocol uses the flow-based concept to identify network traffic, in which the traffic is handled in terms of flows rather than individual packets. As defined in [47], the flows are a packets sequence passing an observation point in the network during a specific time interval. All packets in a flow have common properties such as transport protocol, IP addresses, and ports from both source and destination.

Previous works in the literature collected information from the network data using time intervals between 1 and 5 minutes [48, 49]. However, this analysis interval has been reduced due to the high transmission rates reached in the current network scenario. The new solutions present in the literature have used intervals analysis near-real-time [50, 51]. In that way, we reduce the time interval analysis to one-second. Every second, the proposed system requests and analyzes the IP flow records collected from each switch. This interval enables the system to react fast against anomaly events by reducing the time response.

For all time t, the controller sends requests to collect IP flow records for switches belonging to the network through OpenFlow's Read-State messages. After each switch receives this message, they send a response containing each flow record stored in their forwarding table. This process is represented in Figure 2. The collected data can be defined as a set $\beta_t = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, where each α_i is a flow record that is composed by the following features presented in Table 1:

Table 1: Collected flow features.

Flow Feature	Description
$x^{bits}_{\in\alpha_i}$	The amount of bits belongs to the flow α_i
$x^{packets}_{\in\alpha_i}$	The amount of packets belongs to the flow α_i
$x^{srcIP}_{\in\alpha_i}$	Source IP address
$x^{srcPort}_{\in\alpha_i}$	Source port number
$x^{dstIP}_{\in\alpha_i}$	Destination IP address
$x^{dstPort}_{\in\alpha_i}$	Destination port number

The collected flow features can be classified as quantitative (bits and packets) and qualitative (source IP address, destination IP address, source, and destination ports). The quantitative flow features provide information about traffic volume, which is essential to understand the amount of traffic transported on the network. On the other hand, the qualitative flow features allow us to understand which devices communicate and which applications are being accessed.

3.2. Data Processing

The previous module's IP flow records need to be processed to extract specific characteristics that assist in the anomaly detection approach. The data processing module is responsible for grouping the flow attributes in each analysis interval. Every second, the following attributes are processed:

- *bits/s*
- *packets/s*
- *Source IP Entropy*

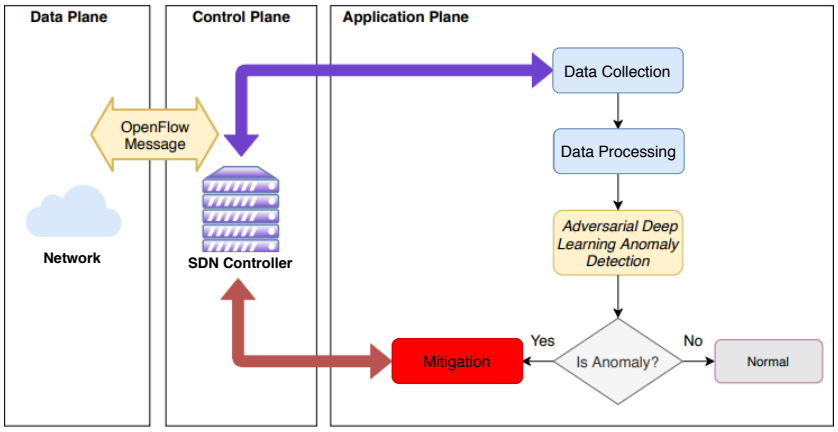

Figure 1: Proposed system architecture using Generative Adversarial Network (GAN) framework.

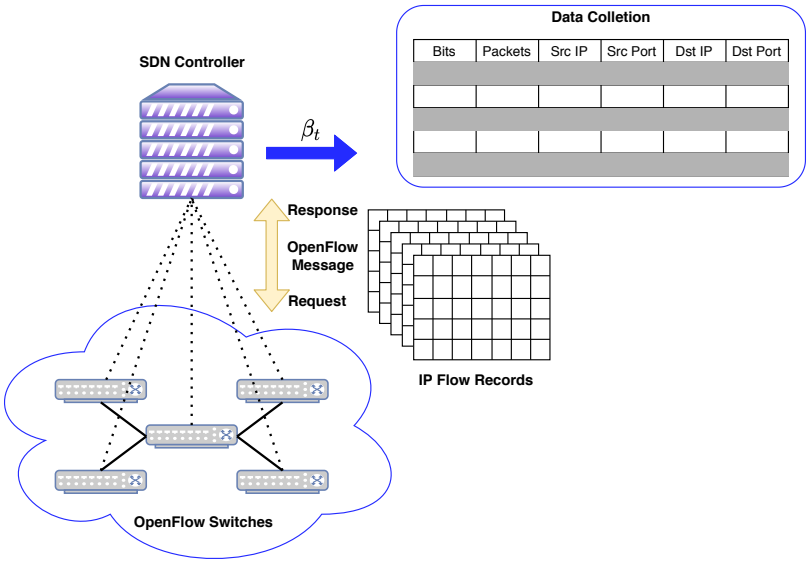

Figure 2: Collecting Flow-Features from OpenFlow Switches.

- *Source Port Entropy*
- *Destination IP Entropy*
- *Destination Port Entropy*

These flow features were extensively analyzed and employed in previous works present in the literature [52]. They have been used in network traffic characterization of high speed, and the outcomes achieved for anomaly detection have been effective.

The rate of bits and packets per second, the quantitative features, are processed by:

$$bits/s = \sum_{j=1}^{|\beta_t|} x_j^{bits} \quad (1)$$

$$packets/s = \sum_{j=1}^{|\beta_t|} x_j^{packets} \quad (2)$$

IP addresses and port numbers are categorical features. However, the detection module applies a deep neural network to detect anomalies events. The deep neural approaches require that input variables are numbers. Thus, the qualitative features, IP and ports, must be casting to numerical type. A simple transformation is to group these data by calculating the entropy. In this module, we applied the Shannon Entropy [53], which highlights the concentration or dispersion degree of the traffic features during an analyzed time interval. Given a set of a qualitative feature, such as $x^{srcIP} = \{x_1, x_2, \ldots, x_n\}$, in which a x_i represents the occurrences number of source IP address i at a given time interval t. The Shannon Entropy $H(\cdot)$ for the flow feature x^{srcIP} is defined as:

$$H(x^{srcIP}) = -\sum_{i=1}^{n} \left(\frac{x_i}{S}\right) log_2 \left(\frac{x_i}{S}\right), \quad (3)$$

in which $S = \sum_{i=1}^{n} x_i$ is the sum of all occurrences of the elements present on the analyzed time interval t. The entropy calculation for the other qualitative attributes (like $H(x^{srcPort})$,

$H(x^{dstIP})$, and $H(x^{dstPort})$) is calculated in the same manner as presented in the Eq. (3).

Information on the degree of concentration or dispersion of a given flow feature can help during the anomaly detection phase. For instance, under a DDoS attack occurrence, the destination IP address and destination Port number distribution may become concentrated due to the high number of connections requested by the attackers. The concentration and dispersion of the flow attributes affected during the occurrence of a DDoS attack is illustrated in Figure 3. This figure presents the analysis results of Source IP Entropy, Destination IP Entropy, Source Port Entropy, and Destination Port Entropy for a day of network traffic used to evaluate our approach. The blue intervals represent the entropy measured during the normal behavior of the network. As can be seen, the entropy value remains continuous, without significant variations in the normal ranges. On the contrary, the intervals in red represent the occurrence of DDoS attacks. There is a concentration in these intervals of the source and destination IP addresses and the destination port, as shown. On the other hand, there is a dispersion of the source port's entropy because multiple attackers use random source ports.

3.3. Adversarial Deep Learning Anomaly Detection

The Adversarial Deep Learning Anomaly Detection step aims to detect and identify malicious agents' activities that can cause anomalous behavior. In this module, it is necessary to use techniques capable of differentiating the normal traffic operation from possible attacks against the network. Recently deep neural networks (DNN) have been applied in anomaly detection systems [9, 13]. These systems have outperformed all the other classical machine learning systems. Though DNN approaches suffer against adversarial examples. To address this problem, we apply adversarial training through Generative Adversarial Network (GAN) framework.

GAN is a framework proposed by Goodfellow *et al.* [34], which simultaneously trains two models through an adversarial process. The GAN framework comprises two neural networks models, a generative model (G) and a discriminative model (D). Both models compete against each other in an adversarial way. Considering that, this competition consists of a minimax two-player game according to the game theory scenario. The G model generates fakes samples from a noise distribution similar to the original dataset. On the other hand, the D model determines the probability that the samples belong to the original dataset.

The G model builds a mapping from a prior noise p_z to a data space $G(z)$ and learns a generative distribution p_g over the data x. In order to increase the error rate of D, the fake samples must be as similar as possible to the real distribution p_{data}. The Eq. (4) represents the objective function of G model:

$$min\frac{1}{2}\,\mathbb{E}_{z\sim p_z}\,log(1-D(G(z))) \tag{4}$$

where $z\sim p_z$ is data from the distribution generated by G and $D(G(z))$ indicates the probability of D determining the data generated by G. The G model achieves its training minimizing Eq. (4), which means that the D model predicts the generated data $G(z)$ with a high probability. The D model aims to classify whether a sample is a real data or a generated data. Thus, the objective function of D is defined in Eq. (5):

$$Max\frac{1}{2}\,\mathbb{E}_{x\sim pdata}\,logD(x)+\frac{1}{2}\,\mathbb{E}_{z}\,log(1-D(G(z))) \tag{5}$$

where $D(x)$ determines the probability of the real data and $x\sim p_{data}$ is the data from the original distribution. The D model effectiveness is achieved maximizing the Eq. (5). So that, considering the conflict between the two models, the GAN framework can be defined as minimax game. Both models continuously improve their effectiveness until an equilibrium is reached. Eq. (6) formalized the minimax game:

$$minMax\frac{1}{2}\,\mathbb{E}_{x\sim pdata}\,logD(x)+\frac{1}{2}\,\mathbb{E}_{z}\,log(1-D(G(z))) \tag{6}$$

As previously identified, the GAN framework includes the generator model (G) and the discriminator model (D). Both models implemented in our approach are Deep Neural Networks (DNNs) structure. Neural networks with multiple hidden layers are qualified as Deep Neural Networks [36]. DNNs enable hierarchically extracting knowledge abstraction and learning characteristics and patterns from the raw network data. Besides, DNNs give more representation of input data to improve the rate detection of complex attacks in a high-speed network environment, mainly DDoS attacks [54, 55].

The generator (G) was implemented with multiple fully-connected layers (Dense) and a linear output layer. Table 2 shows the G model structure. We also implemented multiple fully-connected layers for the discriminator (D), but we implemented a sigmoid layer for the output layer, in which output is the classification of the network's behavior in that analysis interval. In that manner, our approach classifies the network traffic as normal or DDoS attack. Besides, the discriminator (D) during the adversarial training learns the network's normal behavior, which is advantageous for detecting zero-day attacks. Table 3 details the D model's layers.

Table 2: Structure of the Generator (G) model.

#	Layer	Neurons	Activation
1	Fully-connected (Dense)	6	ReLU
2	Fully-connected (Dense)	10	ReLU
3	Fully-connected (Dense)	8	ReLU
4	Fully connected (Dense)	6	Linear

Table 3: Structure of the Discriminator (D) model.

#	Layer	Neurons	Activation
1	Fully-connected (Dense)	12	ReLU
2	Fully-connected (Dense)	10	ReLU
3	Fully-connected (Dense)	6	ReLU
4	Fully connected (Dense)	1	Sigmoid

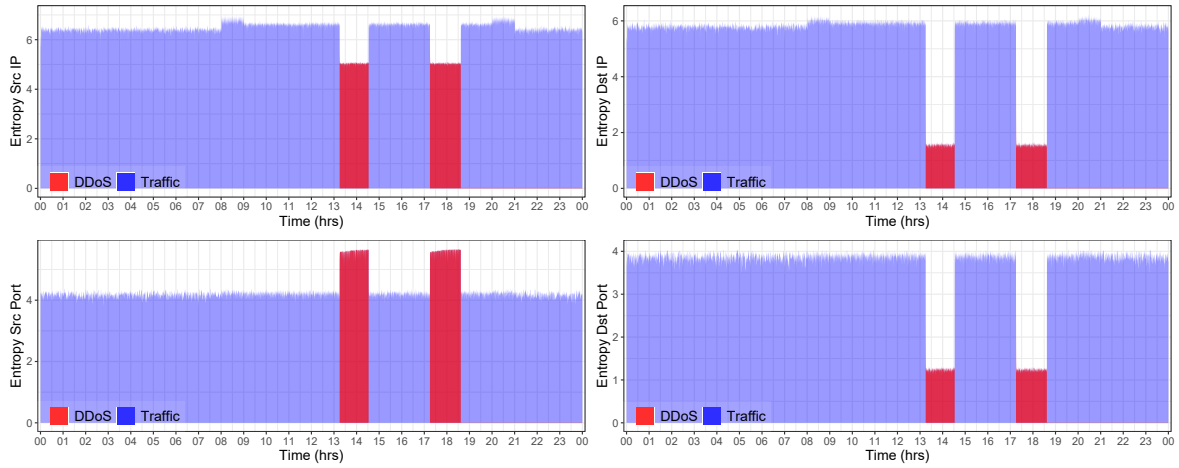

Figure 3: Flow features entropy analysis during DDoS attacks.

The flowchart present in Figure 4 illustrated the adversarial training process used in our approach. The first step is training the D model for d_{epoch}. During this step, a minibatch of m samples from the training dataset are randomly sampled. The training dataset is composed of normal traffic and DDoS attacks. Then, the G model generates adversarial a examples from z random noise. Both m and a sets are used to training the D model. The weights of the D and G models are updated according to its stochastic gradient. The stochastic gradients of D and G models are defined in Eq. (7) and Eq. (8), respectively.

$$\nabla_{\theta_D} \frac{1}{m} \sum_{i=1}^{m} \left[logD\left(m^{(i)}\right) + log\left(1 - D(G(z^{(i)}))\right)\right] \quad (7)$$

$$\nabla_{\theta_G} \frac{1}{z} \sum_{i=1}^{z} log\left(1 - D(G(z^{(i)})\right) \quad (8)$$

The adversarial training process of generating and training is repeated until the D model can detect the fake samples or a number max of training iterations t. This process is illustrated as follows in Algorithm 1:

Algorithm 1 GAN Adversarial training steps

Require: Generator model G; Discriminator model D; Training dataset

1: **while** t iterations training or stop condition not met **do**
2: **for** d_{epoch} **do**
3: Sample minibatch of m from the training dataset
4: Generates a adversarial examples from G
5: Update the discriminator by ascending its stochastic gradient by Eq. (7)
6: **end for**
7: Sample minibatch of z samples from noise prior $z \sim p_z$
8: Update the generator by descending its stochastic gradient by Eq. (8)
9: **end while**
10: **return** G and D;

3.4. Mitigation

The mitigation module is triggered after a DDoS attack is detected by the Adversarial Deep Learning Anomaly Detection. This module aims to take countermeasures to minimize the damages caused during an attack. In previous work [56], we have proposed a mitigation anomaly approach that achieved efficient outcomes. In this manner, we extended the proposed mitigation approach to this module.

Basically, the approach described in [56] is **Event-Condition-Action** (ECA) model-based, in which the **Event** refers to a specific anomaly associated with a set of rules. The Condition describes the rules where a specific anomaly event took place. Lastly, the **Action** is a countermeasure taking against an anomaly event.

Some network anomalies (e.g., Flash crowd) have characteristics similar to an attack DDoS attack, but they are requests from legitimate users. To overcome this vulnerability, a *Safe List* mechanism was implemented, which maintains a list of flow features from legitimate users.

The mitigation process is summarized as follow in Algorithm 2.

Algorithm 2 Mitigation Process.

Require: Suspect flows β_t

1: Identify the suspect flows based on IP addresses and ports that make the analysis interval anomalous
2: Identify the destination IP address which receives the most flows
3: Identify in those flows the attackers' IP address which have the same destination port
4: **if** IPs e ports are on the *Safe List* **then**
5: Forward packets
6: **else**
7: Drop packets
8: **end if**

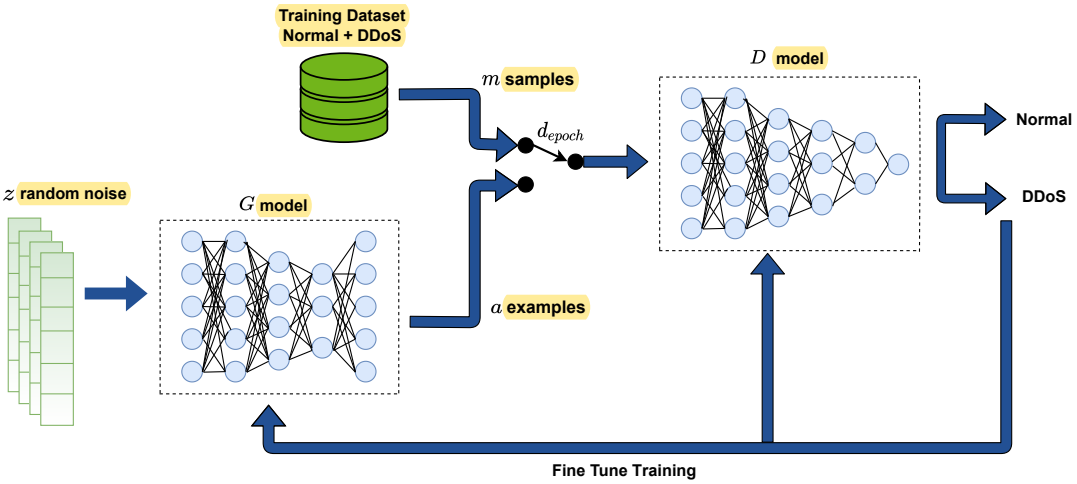

Figure 4: GAN training structure.

4. Experimental Results and Discussion

To evaluate the performance of the proposed system, we applied tests in different scenarios and compared it with some other systems present in literature. We used the Python language with development libraries for Deep Learning application Keras [57] and TensorFlow [58]. During the training step the parameters were set as, Dropout with rate of 0.2 to prevent overfitting [59]; the loss function Binary Crossentropy that is a classical loss function used in binary classification tasks; the default learning rate equals to 0.001 defined in Keras [57], and optimizer was set as Adam [60], which is an adaptive learning rate optimization algorithm for training deep neural networks. The experiments were made in an environment with the following configuration: Intel Core i5 2.21 GHz, 8 GB RAM, Windows 10 system.

The first test scenario contains data from the emulation of an SDN network with a high transmission rate and many devices connected, totaling 128 hosts. In this scenario, it has two periods of UDP DDoS attacks with different intensities. In the second scenario, we applied the public dataset CICDDoS 2019 [61] from the Canadian Institute for Cybersecurity. This dataset contains 28 normal profile network behavior and up-to-date types of DDoS attacks.

4.1. Evaluation Metric

To analyze the proposed system's performance, we adopted the following classic metrics: Accuracy, Precision, Recall, and F1 Score. Accuracy metric indicates the percentage of intervals correctly classified. Precision presents the percentage of intervals classified as DDoS, which are DDoS. Recall shows how effective the model is in identifying DDoS intervals related to all the intervals. The definition of these metrics can be given by:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \times 100\% \quad (9)$$

$$Precision = \frac{TP}{TP + FP} \times 100\% \quad (10)$$

$$Recall = \frac{TP}{TP + FN} \times 100\% \quad (11)$$

where TP, TN, FP, and FN mean True Positive, True Negative, False Positive, and False Negative, respectively.

Finally, F1 Score is the harmonic mean between recall and precision, providing an appropriate score to the system performance. Its statistical formulation is given by Eq. (12):

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (12)$$

4.2. Parameters Evaluation

During the GAN framework training phase we evaluated some parameters, such as minibatch size and the number of d_{epoch} for training the discriminator before updating the generator. These parameter were evaluated to finding a point of equilibrium between the generator and discriminator.

To evaluate the minibatch size, we used 32, 64, 128, 256, 512, and 1024 samples. The number of epochs and accuracy rate were evaluated to obtain the convergence value. Figure 5 illustrates the results of this test. A minibatch size of 32 reached better accuracy rate first, but it decreased after 257 epochs. As observed, the best result reached was using a minibatch size of 64, in which the number of epochs required for convergence was 157.

The next parameter evaluated was the number of d_{epoch} for alternating and updating the generator and the discriminator. This test is detailed in Figure 6. According to the information presented in [33], small values can reach better results. In that manner, the estimation of the referred parameter d_{epoch} was applied between 1 to 3. The number of epochs and accuracy rate were evaluated to obtain the appropriate value of d_{epoch}. As the results achieved in Figure 6, the number of d_{epoch} that reached the appropriate convergence was equal to 2, in which the number of epochs required for training were 157. The other values of d_{epoch} 1 and 3 are unstable and do not converge.

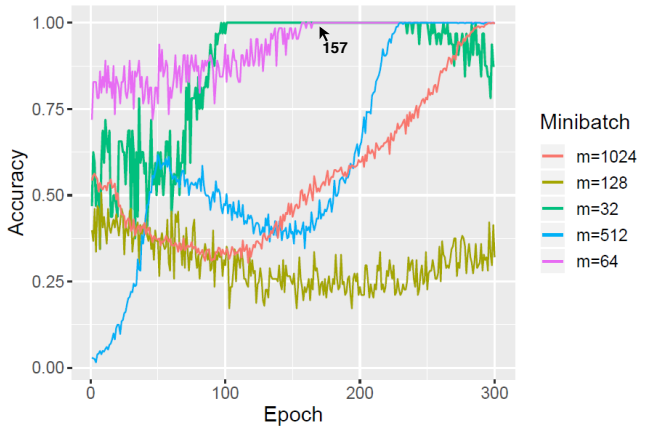

Figure 5: Minibatch size parameter evaluation.

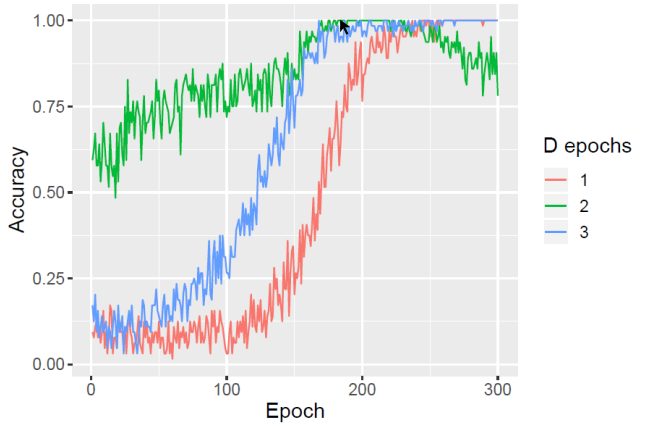

Figure 6: Estimation of the parameter d_{epoch}.

4.3. Scenario 1: SDN Emulated Environment

In this scenario, we emulated a network topology through the Mininet [62] network emulator. The Mininet emulator is widely applied in developing SDN solutions due to the facility for implementing realistic virtual SDN environments composed of controllers, hosts, links, and switches on one virtual machine. The network arrangement emulated is a star topology in which six switches are connected to a central switch. Every subnetwork contains 20 hosts, totaling 120 hosts, as described in Figure 7. Besides, we used the SDN controller Floodlight [63], a controller based on Java that has been utilized in many SDN applications as a network controller.

We emulated the behavior of a network with high transmission rates for 24 hours. To inject traffic in the emulated network, we used a tool called Scapy [64], a packet manipulation tool for computer networks. This tool provides an emulated network that can be similar to a real network scenario.

Two UDP DDoS flood attacks were carried out during the emulation stage with different intensities and duration time. This type of attack is the common DDoS method used by attackers against network services [8]. The parameters used in the attacks are shown in detail in Table 4. This dataset is available online [1].

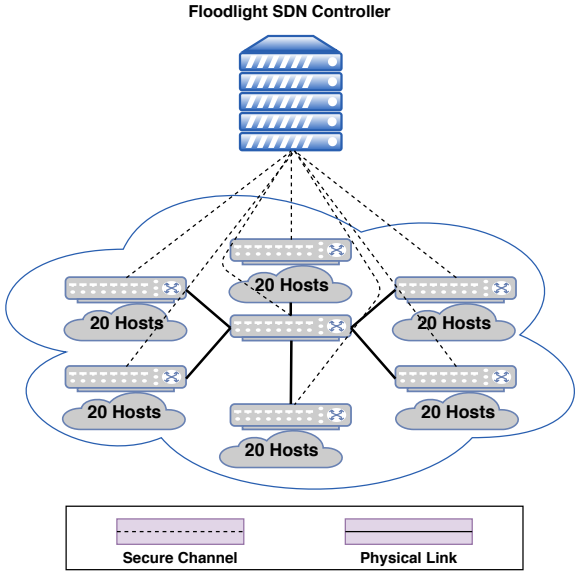

Figure 7: Emulated SDN network on scenario one.

Table 4: UDP DDoS flood parameters.

Type of Attack	Attack Parameters
DDoS #1	Attackers: 15 Attacking IPs: 10.0.0.21 - 10.0.0.35 Victim IP address: 10.0.0.92:2000 Time: 10:10 - 11:20
DDoS #2	Attacking IPs: 10.0.0.45 - 10.0.0.60 Victim IP address: 10.0.0.33:8080 Time: 14:45 - 16:00

We evaluated the GAN framework performance and compared it with other neural networks-based methods present in literature that were also applied to detect DDoS attacks in SDN environments. The methods are the Convolutional Neural Network (CNN) [65], the Long Short-Term Memory (LSTM) [66]; and finally, the classical neural network Multilayer Perceptron (MLP) [54]. These methods have reached accurate outcomes in detecting attacks in SDN, and we compared our system performance with them.

Figure 8 shows the outcomes achieved by each one of the compared methods through the evaluated metrics. Regarding the metrics, the methods GAN, CNN, LSTM, MLP presented values greater than 95%. For *Accuracy*, GAN and CNN obtained similar results, with values of 99.78 and 99.69, respectively. The LSTM reached an accuracy rate of 97.21%, and the MLP reached 95.91%, which means these last two methods were less accurate in classifying the analysis intervals.

The next evaluated metric was the *Precision* rate. GAN achieved a Precision rate of 99.76%, an improvement in 2.19%,

[1] http://www.uel.br/grupos/orion/datasets.html

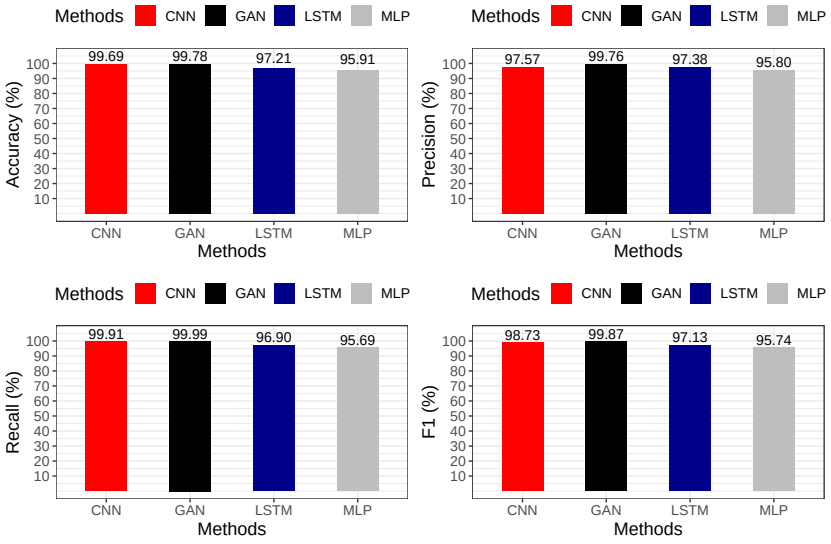

Figure 8: Comparison outcomes between GAN and the compared methods through the evaluated metrics on first scenario.

2.38%, and 4.26% higher than CNN, LSTM, and MLP, respectively. In the following, we evaluated the *Recall* metric. GAN framework also fared better, with a rate of 99.99% for this metric, followed by CNN, LSTM, and MLP methods, reaching rates of 99.91%, 96.90%, and 95.69%, respectively.

Finally, we further examined the robustness of all compared methods using the $F1$ score. GAN reached the best result, with a rate of 99.87%, the remaining ones were CNN, LSTM, and MLP, reaching rates of 98.73%, 97.13%, and 95.74%, respectively.

According to the results achieved for all evaluated metrics, the proposed GAN framework presented an adequate performance in detecting DDoS attacks. Figure 9 presents a radar chart that summarizes all compared methods' performance metrics previously addressed in a single chart. The GAN is closer to the outer circle, which means that closer to 100% the analyzed method fared better for the four evaluated metrics, followed by CNN, LSTM, and MLP.

The analysis intervals detected as attacks by the detection module are reported to the mitigation module. In these intervals, mitigation policies are applied. Figure 10 presents the behavior of the six attributes flow analyzed before and after the mitigation process. The blue bar plot and the green line plot illustrate network traffic before and after applying the mitigation policy. An increase in the bits and packets rates can be noticed before applying the mitigation. After the mitigation, the network traffic behavior tends to go back to its normality due to anomalous packets' discards.

4.4. Scenario 2: CICDDoS 2019 dataset

In this second test scenario, we evaluated the proposed system's performance for detecting different types of DDoS attacks. In order to do so, we applied the public dataset called CICDDoS 2019. This dataset contains 28 normal profile network behaviors and the most up-to-date common types of DDoS attacks. The dataset has been organized by day, one for training and another for testing. The training dataset comprises 12 different modern DDoS attacks such as NTP, DNS, LDAP, MSSQL, NetBIOS, SNMP, SSDP, UDP, UDP-Lag, WebDDoS (ARME), SYN, and TFTP. The testing day contains 6 types of DDoS attacks, including NetBIOS, LDAP, MSSQL, UDP, UDP-Lag, and SYN. In that manner, the DDoS types explored in this scenario are summarized as follow:

- Network Time Protocol (NTP) server functionality are used to overwhelm a targeted network by sending UDP packets with spoofed IP addresses;
- Domain Name System (DNS) is an attack that the attacker overwhelms a particular server in order to disrupt that domain which can be carried out using either TCP or UDP packets;
- Lightweight Directory Access (LDAP) is a protocol used for directory services on corporate networks. The attacker sends small requests to a vulnerable LDAP server to produce amplified replies, which floods the victim;
- Microsoft Structured Query Language (MSSQL) is an attack that makes it possible to execute malicious SQL queries;
- NetBios is an attack that sends many spoofed "Name Release" or "Name Conflict" messages, forcing the machine to remove its own name from its name table and becoming unable to other NetBIOS;
- Simple Network Management Protocol (SNMP) attack is a type of DDoS in which the attacker sends a large number of requests with spoofed IP address to several devices;
- Simple Service Discovery Protocol (SSDP) attack is a type of DDoS that uses Universal Plug and Play (UPnP)

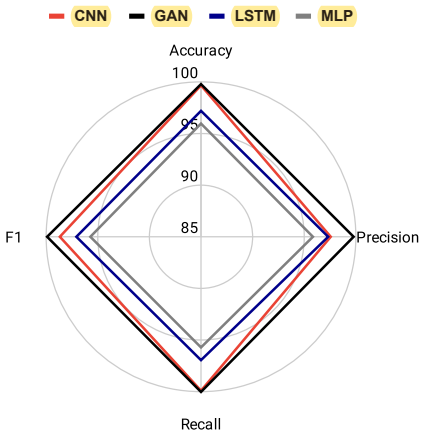

Figure 9: Radar chart results for the evaluated methods on first scenario.

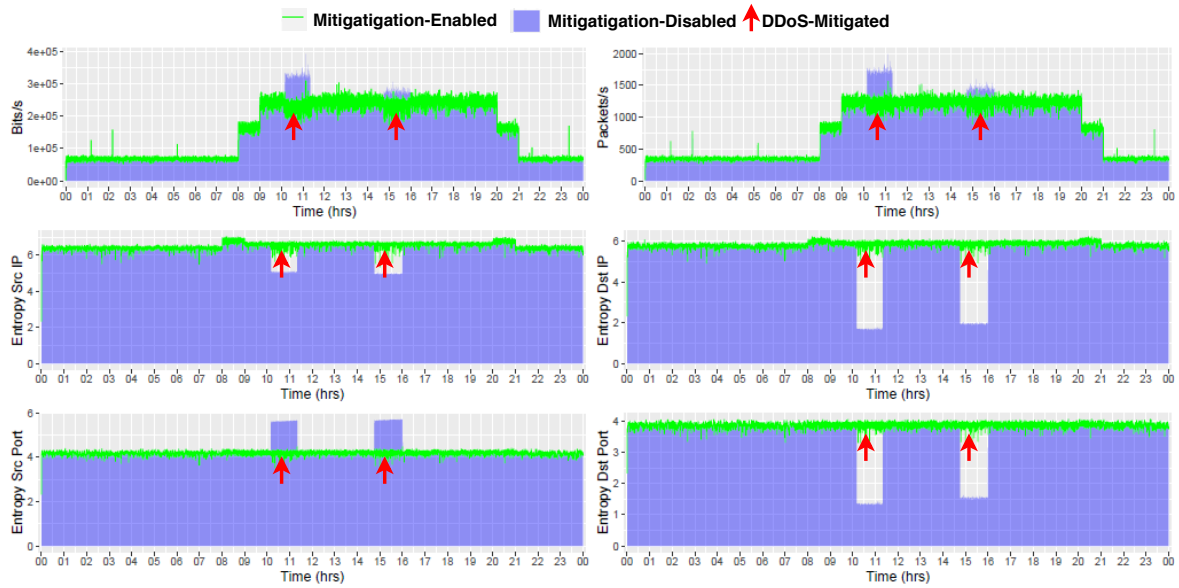

Figure 10: Flow features behavior before and after the mitigation process on scenario one

networking protocols to send a massive amount of traffic to a targeted victim;

- User Datagram Protocol (UDP) packets are sent by an attacker to overwhelm the target's ability to process and respond to it;
- UDP-lag is a type of attack that interrupts the connection between the target client and the server;
- WebDDoS (ARME) attack is a DDoS to crash the website or slow it by flooding the network with multiple fake requests to the target;
- SYN attack is a DDoS in which packets with spoofed IP addresses are sent to overwhelm all available ports to a targeted server;
- Trivial File Transfer Protocol (TFTP) attack occurs when an attacker overflows the buffer server.

GAN efficiency measurements were performed using the same classical metrics as carried out in the first scenario. We also compared the results achieved with the methods CNN [65], LSTM [66], and MLP [54]. Figure 11 illustrates the measurement results of the metrics obtained for each of them.

As it can be noticed from Figure 11, the methods GAN and CNN reached similar results for the *Accuracy* metric, with rates of around 94% (GAN reached 0.3% greater than CNN). The other two methods, LSTM and MLP, reached Accuracy rates of 90.29% and 92.12%, respectively.

Regarding the *Precision* metric, the CNN method fared relatively better than the other compared methods, reaching a rate of 96.32%, followed by GAN, MLP, and LSTM with rates of 94.08%, 92.12%, and 90.12%, respectively. For the *Recall* met-

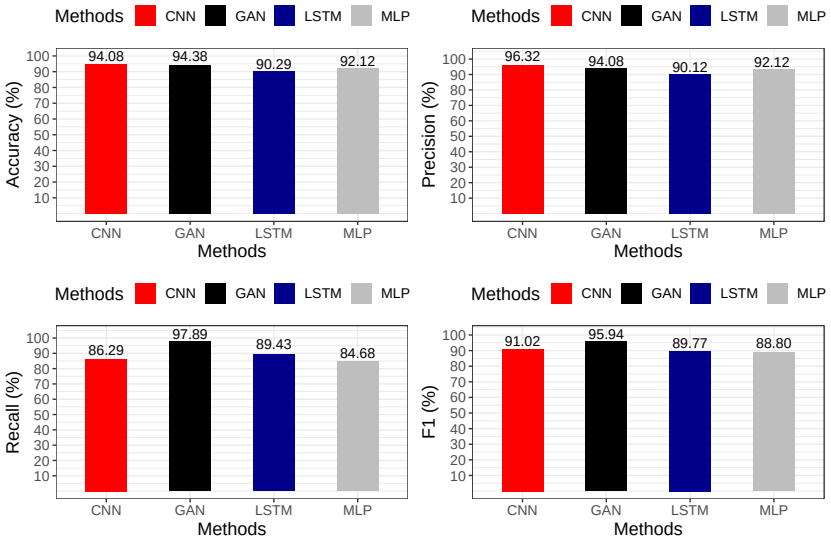

Figure 11: Comparison outcomes between GAN and the compared methods through the evaluated metrics on second scenario.

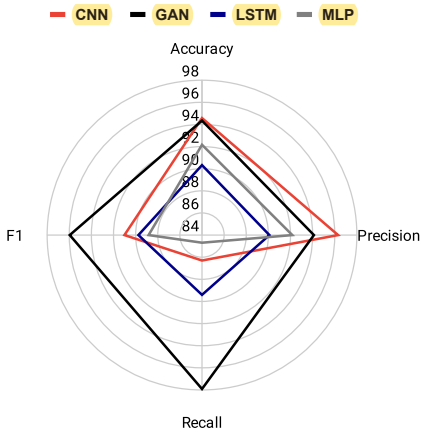

Figure 12: Radar chart results for the evaluated methods on second scenario.

ric, the GAN obtained superior performance, achieving a rate of 97.89% for this metric, a rate improvement of 8.46%, 11.6%, and 13.21% higher than LSTM, CNN, and MLP, respectively.

As mentioned in the previous scenario, we also used the F1 score to determine how precise and robust the methods are in detecting different types of DDoS attacks. Regarding the $F1$ score, it is clear that the GAN framework yielded better outcomes than the other methods, with a rate of 95.54%. The methods CNN, LSTM, MLP fared similar rates, with 91.02%, 89.77%, and 88.80%, respectively.

We summarized all outcomes measurements achieved by the tested methods in a radar chart. These outcomes are illustrated in Figure 12. The methods compared in this scenario obtained significantly lower results compared to the first scenario. The main reason for this difference is the number of attacks evaluated, making the detection process difficult. However, GAN achieved superior results due to its training process that used an adversary training approach, making it less sensitive to different attacks. As the results gained in the first scenario, the GAN framework also reached the best outcomes on average. The proposed system provides an efficient approach capable of detecting different kinds of DDoS attacks.

In this scenario, we also evaluated the mitigation performance against several DDoS attacks. The anomaly detection module triggered the anomalous intervals to the mitigation module, in which the module automatically applied the DDoS countermeasure. Figure 13 shows the network traffic behavior for each flow feature. The traffic before the mitigation is represented in the blue area, and the green line shows the traffic after applying the mitigation policy. As it can be seen in Figure 13 the flow features variations were inhibited after the mitigation module is triggered. Through this analysis, the mitigation module could mitigate the DDoS attacks successfully, and the normal traffic has been maintained.

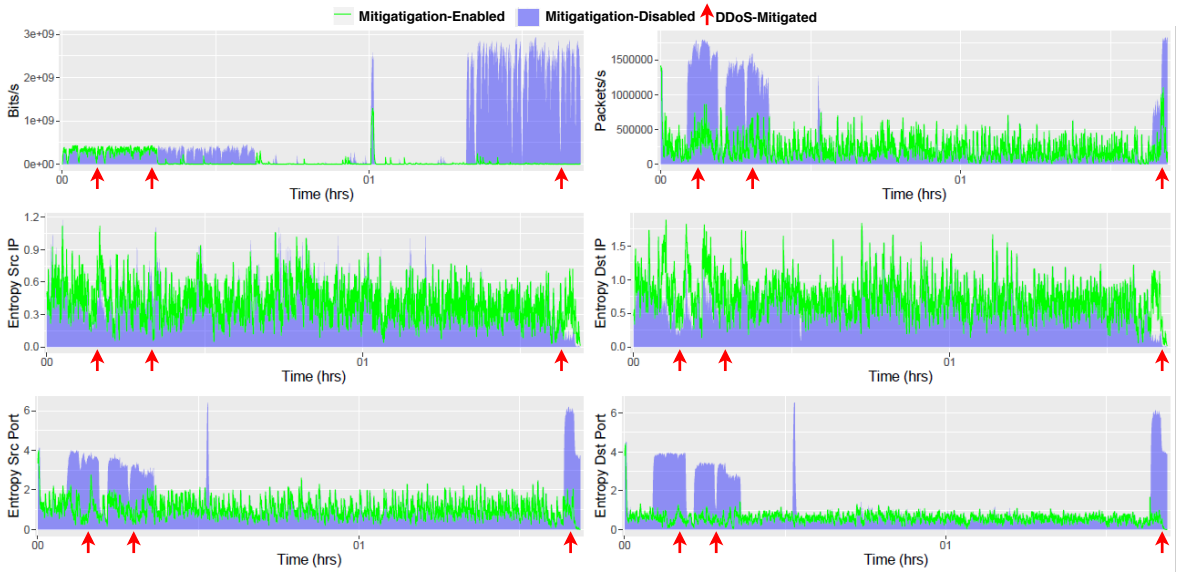

Figure 13: Flow features behavior before and after the mitigation process on CICDDoS 2019 dataset.

5. Conclusion

In this work, we proposed a system detection and defense against DDoS attacks in SDN environments. This system comprises four integrated modules that provide tools for collecting, processing, detecting, and mitigating attacks. We used the Generative Adversarial Network (GAN) and applied adversarial training to make the system less sensitive to adversarial attacks.

We compared the proposed system's outcomes with different neural network methods present in literature for detecting DDoS in SDN, such as CNN, LSTM, and MLP. During the test stage, the methods were submitted to two test scenarios. The first scenario emulated a real SDN environment with many hosts and high transmission rates, using the Mininet emulator and the Floodlight controller. In the second scenario, we evaluated the system through the recent public dataset CICDDoS 2019, which contains the most 12 up-to-date common types of DDoS attacks such as NTP, DNS, LDAP, MSSQL, NetBIOS, SNMP, SSDP, UDP, UDP-Lag, WebDDoS (ARME), SYN, and TFTP.

In the first scenario, we evaluated the proposed system for detecting UDP flood attacks in an SDN network environment with high transmission rates. The network traffic was analyzed near-real-time, which was performed in one-second time intervals. The results gained by the prosed system have fared better than the other compared methods in that test environment. According to the evaluated metrics, the system was able to detect and defend against almost DDoS performed.

In the second scenario, we tested the system performance for detecting DDoS attacks against different applications. In that scenario, the compared methods were less accurate in detecting these attacks. On the other side, we optimized our system using the GAN framework through adversarial training that provided adversarial examples to enhance its ability to defend against them, making the system more accurate in identifying the different DDoS attacks. In both scenarios, the proposed system obtained results superior to the other compared methods, allowing its application against several DDoS attacks on SDN environment.

The GAN framework has shown its potential for use in environments susceptible to different threats due to its adversarial training that makes the system less sensitive to adversarial attacks. In conclusion, the GAN framework improved the performance indicators evaluated in this work and showed the powerful capacity to detect novel attacks.

In our future work, we intend to explore the impact of other deep neural methods (e.g., Gated-Recurrent Unit, Stacked Auto-Encoder) on the General and Discriminator. We can consider applying tests, increasing the number of hosts and switches on the emulated scenario. Besides, we will pay efforts to detect and classify other attacks in a multiclass classification approach.

Acknowledgement

This work has been partially supported by the National Council for Scientific and Technological Development (CNPq) of Brazil under Grant of Project 310668/2019-0 and by SETI/Fundação Araucária due to the concession of scholarships; by the "Ministerio de Economía y Competitividad" in the "Programa Estatal de Fomento de la Investigación Científica y Técnica de Excelencia, Subprograma Estatal de Generación de Conocimiento" within the project under Grant TIN2017-84802-C2-1-P.

References

[1] X. Zhang, L. Cui, K. Wei, F. P. Tso, Y. Ji, W. Jia, A survey on stateful data plane in software defined networks, Computer Networks (2020) 107597doi:https://doi.org/10.1016/j.comnet.2020.107597. URL http://www.sciencedirect.com/science/article/pii/S1389128620312305

[2] T. Das, V. Sridharan, M. Gurusamy, A survey on controller placement in sdn, IEEE Communications Surveys Tutorials 22 (1) (2020) 472–503. doi:10.1109/COMST.2019.2935453.

[3] O. Salman, I. Elhajj, A. Chehab, A. Kayssi, Iot survey: An sdn and fog computing perspective, Computer Networks 143 (2018) 221 – 246. doi:https://doi.org/10.1016/j.comnet.2018.07.020. URL http://www.sciencedirect.com/science/article/pii/S1389128618305395

[4] S. Garg, K. Kaur, N. Kumar, J. J. P. C. Rodrigues, Hybrid deep-learning-based anomaly detection scheme for suspicious flow detection in sdn: A social multimedia perspective, IEEE Transactions on Multimedia 21 (3) (2019) 566–578. doi:10.1109/TMM.2019.2893549.

[5] C. Gkountis, M. Taha, J. Lloret, G. Kambourakis, Lightweight algorithm for protecting sdn controller against ddos attacks, in: 2017 10th IFIP Wireless and Mobile Networking Conference (WMNC), 2017, pp. 1–6. doi:10.1109/WMNC.2017.8248858.

[6] K. S. Sahoo, D. Puthal, M. Tiwary, J. J. Rodrigues, B. Sahoo, R. Dash, An early detection of low rate ddos attack to sdn based data center networks using information distance metrics, Future Generation Computer Systems 89 (2018) 685–697. doi:https://doi.org/10.1016/j.future.2018.07.017. URL https://www.sciencedirect.com/science/article/pii/S0167739X18309543

[7] M. P. Singh, A. Bhandari, New-flow based ddos attacks in sdn: Taxonomy, rationales, and research challenges, Computer Communications 154 (2020) 509 – 527. doi:https://doi.org/10.1016/j.comcom.2020.02.085.

[8] O. Yurekten, M. Demirci, Sdn-based cyber defense: A survey, Future Generation Computer Systems 115 (2021) 126 – 149. doi:https://doi.org/10.1016/j.future.2020.09.006. URL http://www.sciencedirect.com/science/article/pii/S0167739X20303277

[9] Ömer KASIM, An efficient and robust deep learning based network anomaly detection against distributed denial of service attacks, Computer Networks 180 (2020) 107390. doi:https://doi.org/10.1016/j.comnet.2020.107390. URL http://www.sciencedirect.com/science/article/pii/S1389128620304114

[10] N. Garcia, T. Alcaniz, A. González-Vidal, J. B. Bernabe, D. Rivera, A. Skarmeta, Distributed real-time slowdos attacks detection over encrypted traffic using artificial intelligence, Journal of Network and Computer Applications 173 (2021) 102871. doi:https://doi.org/10.1016/j.jnca.2020.102871. URL http://www.sciencedirect.com/science/article/pii/S1084804520303362

[11] N. Moustafa, J. Hu, J. Slay, A holistic review of network anomaly detection systems: A comprehensive survey, Journal of Network and Computer Applications 128 (2019) 33 – 55. doi:https://doi.org/10.1016/j.jnca.2018.12.006.

[12] A. Aldweesh, A. Derhab, A. Z. Emam, Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues, Knowledge-Based Systems 189 (2020) 105124. doi:https://doi.org/10.1016/j.knosys.2019.105124. URL http://www.sciencedirect.com/science/article/pii/S0950705119304897

[13] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, H. Janicke, Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study, Journal of Information Security and Applications 50 (2020) 102419. doi:https://doi.org/10.1016/j.jisa.2019.102419.

[14] D.-T. Hoang, H.-J. Kang, A survey on deep learning based bearing fault diagnosis, Neurocomputing 335 (2019) 327 – 335. doi:https://doi.org/10.1016/j.neucom.2018.06.078. URL http://www.sciencedirect.com/science/article/pii/S0925231218312657

[15] A. M. Ozbayoglu, M. U. Gudelek, O. B. Sezer, Deep learning for financial applications : A survey, Applied Soft Computing 93 (2020) 106384. doi:https://doi.org/10.1016/j.asoc.2020.106384. URL http://www.sciencedirect.com/science/article/pii/S1568494620303240

[16] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[17] D. H. Ackley, G. E. Hinton, T. J. Sejnowski, A learning algorithm for boltzmann machines, Cognitive Science 9 (1) (1985) 147 – 169. doi:https://doi.org/10.1016/S0364-0213(85)80012-4.

[18] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation 9 (8) (1997) 1735–1780, dOI:10.1162/neco.1997.9.8.1735. doi:10.1162/neco.1997.9.8.1735.

[19] J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proceedings of the National Academy of Sciences 79 (8) (1982) 2554–2558. arXiv:https://www.pnas.org/content/79/8/2554.full.pdf, doi:10.1073/pnas.79.8.2554.

[20] G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507. doi:10.1126/science.1127647.

[21] S. Li, W. Song, L. Fang, Y. Chen, P. Ghamisi, J. A. Benediktsson, Deep learning for hyperspectral image classification: An overview, IEEE Transactions on Geoscience and Remote Sensing 57 (9) (2019) 6690–6709. doi:10.1109/TGRS.2019.2907932.

[22] V. Carletti, A. Greco, G. Percannella, M. Vento, Age from faces in the deep learning revolution, IEEE Transactions on Pattern Analysis and Machine Intelligence 42 (9) (2020) 2113–2132. doi:10.1109/TPAMI.2019.2910522.

[23] T. T. Nguyen, N. D. Nguyen, S. Nahavandi, Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications, IEEE Transactions on Cybernetics 50 (9) (2020) 3826–3839. doi:10.1109/TCYB.2020.2977374.

[24] L. Lei, Y. Tan, K. Zheng, S. Liu, K. Zhang, X. Shen, Deep reinforcement learning for autonomous internet of things: Model, applications and challenges, IEEE Communications Surveys Tutorials 22 (3) (2020) 1722–1760. doi:10.1109/COMST.2020.2988367.

[25] C. Athanasiadis, E. Hortal, S. Asteriadis, Audio–visual domain adaptation using conditional semi-supervised generative adversarial networks, Neurocomputing 397 (2020) 331 – 344. doi:https://doi.org/10.1016/j.neucom.2019.09.106.

[26] I. Rasheed, F. Hu, L. Zhang, Deep reinforcement learning approach for autonomous vehicle systems for maintaining security and safety using lstm-gan, Vehicular Communications 26 (2020) 100266. doi:https://doi.org/10.1016/j.vehcom.2020.100266.

[27] P. Wang, E. Fan, P. Wang, Comparative analysis of image classification algorithms based on traditional machine learning and deep learning, Pattern Recognition Lettersdoi:https://doi.org/10.1016/j.patrec.2020.07.042.

[28] M. Pikuliak, M. Šimko, M. Bieliková, Cross-lingual learning for text processing: A survey, Expert Systems with Applications (2020) 113765doi:https://doi.org/10.1016/j.eswa.2020.113765.

[29] W. G. Hatcher, W. Yu, A survey of deep learning: Platforms, applications and emerging research trends, IEEE Access 6 (2018) 24411–24432. doi:10.1109/ACCESS.2018.2830661.

[30] X. Yuan, P. He, Q. Zhu, X. Li, Adversarial examples: Attacks and defenses for deep learning, IEEE Transactions on Neural Networks and Learning Systems 30 (9) (2019) 2805–2824. doi:10.1109/TNNLS.2018.2886017.

[31] M. Pawlicki, M. Choraś, R. Kozik, Defending network intrusion detection systems against adversarial evasion attacks, Future Generation Computer Systems 110 (2020) 148 – 154. doi:https://doi.org/10.1016/j.future.2020.04.013. URL http://www.sciencedirect.com/science/article/pii/S0167739X20303368

[32] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, X. Yi, A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability, Computer Science Review 37 (2020) 100270. doi:https://doi.org/10.1016/j.cosrev.2020.100270. URL http://www.sciencedirect.com/science/article/pii/S1574013719302527

[33] X. Zhang, Y. Zhou, S. Pei, J. Zhuge, J. Chen, Adversarial examples detection for xss attacks based on generative adversarial networks, IEEE Access 8 (2020) 10989–10996. doi:10.1109/ACCESS.2020.2965184.

[34] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Proceedings of the 27th International Conference on Neural Information Pro-

cessing Systems - Volume 2, NIPS'14, MIT Press, Cambridge, MA, USA, 2014, p. 2672–2680.

[35] H. Li, S. Zhou, W. Yuan, J. Li, H. Leung, Adversarial-example attacks toward android malware detection system, IEEE Systems Journal 14 (1) (2020) 653–656. doi:10.1109/JSYST.2019.2906120.

[36] K. Grosse, N. Papernot, P. Manoharan, M. Backes, P. McDaniel, Adversarial examples for malware detection, in: S. N. Foley, D. Gollmann, E. Snekkenes (Eds.), Computer Security – ESORICS 2017, Springer International Publishing, Cham, 2017, pp. 62–79. doi:10.1007/978-3-319-66399-9_4.

[37] 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 Communications Surveys Tutorials 18 (1) (2016) 602–622. doi:10.1109/COMST.2015.2487361.

[38] S. Dong, K. Abbas, R. Jain, A survey on distributed denial of service (ddos) attacks in sdn and cloud computing environments, IEEE Access 7 (2019) 80813–80828. doi:10.1109/ACCESS.2019.2922196.

[39] N. Agrawal, S. Tapaswi, Defense mechanisms against ddos attacks in a cloud computing environment: State-of-the-art and research challenges, IEEE Communications Surveys Tutorials 21 (4) (2019) 3769–3795. doi:10.1109/COMST.2019.2934468.

[40] Z. Li, H. Jin, D. Zou, B. Yuan, Exploring new opportunities to defeat low-rate ddos attack in container-based cloud environment, IEEE Transactions on Parallel and Distributed Systems 31 (3) (2020) 695–706. doi:10.1109/TPDS.2019.2942591.

[41] S. Velliangiri, H. M. Pandey, Fuzzy-taylor-elephant herd optimization inspired deep belief network for ddos attack detection and comparison with state-of-the-arts algorithms, Future Generation Computer Systems 110 (2020) 80 – 90. doi:https://doi.org/10.1016/j.future.2020.03.049.
URL http://www.sciencedirect.com/science/article/pii/S0167739X19332388

[42] G. S. Kushwah, V. Ranga, Voting extreme learning machine based distributed denial of service attack detection in cloud computing, Journal of Information Security and Applications 53 (2020) 102532. doi:https://doi.org/10.1016/j.jisa.2020.102532.
URL http://www.sciencedirect.com/science/article/pii/S2214212619304016

[43] S. Akcay, A. Atapour-Abarghouei, T. P. Breckon, Ganomaly: Semi-supervised anomaly detection via adversarial training, in: C. V. Jawahar, H. Li, G. Mori, K. Schindler (Eds.), Computer Vision – ACCV 2018, Springer International Publishing, Cham, 2019, pp. 622–637. doi:10.1007/978-3-030-20893-6_39.

[44] N. Martins, J. M. Cruz, T. Cruz, P. Henriques Abreu, Adversarial machine learning applied to intrusion and malware scenarios: A systematic review, IEEE Access 8 (2020) 35403–35419. doi:10.1109/ACCESS.2020.2974752.

[45] J. Singh, S. Behal, Detection and mitigation of ddos attacks in sdn: A comprehensive review, research challenges and future directions, Computer Science Review 37 (2020) 100279. doi:https://doi.org/10.1016/j.cosrev.2020.100279.

[46] J. C. Correa Chica, J. C. Imbachi, J. F. Botero Vega, Security in sdn: A comprehensive survey, Journal of Network and Computer Applications 159 (2020) 102595. doi:https://doi.org/10.1016/j.jnca.2020.102595.

[47] P. Aitken, B. Claise, B. Trammell, Specification of the ip flow information export (ipfix) protocol for the exchange of flow information, Tech. Rep. 7011, RFC Editor (2013).
URL https://tools.ietf.org/html/rfc7011

[48] M. Proença Jr., C. Coppelmans, M. Bottoli, A. Alberti, L. S. Mendes, The Hurst Parameter for Digital Signature of Network Segment, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 772–781. doi:10.1007/978-3-540-27824-5_103.

[49] L. Carvalho, T. Abrao, L. Mendes, M. Proença Jr., An ecosystem for anomaly detection and mitigation in software-defined networking, Expert Systems with Applications 104 (2018) 121 – 133. doi:https://doi.org/10.1016/j.eswa.2018.03.027.
URL http://www.sciencedirect.com/science/article/pii/S0957417418301726

[50] S. Garg, K. Kaur, N. Kumar, G. Kaddoum, A. Y. Zomaya, R. Ranjan, A hybrid deep learning-based model for anomaly detection in cloud datacenter networks, IEEE Transactions on Network and Service Management 16 (3) (2019) 924–935. doi:10.1109/TNSM.2019.2927886.

[51] Y. Zhou, G. Cheng, S. Jiang, M. Dai, Building an efficient intrusion detection system based on feature selection and ensemble classifier, Computer Networks 174 (2020) 107247. doi:https://doi.org/10.1016/j.comnet.2020.107247.
URL http://www.sciencedirect.com/science/article/pii/S1389128619314203

[52] E. H. M. Pena, S. Barbon, J. J. P. C. Rodrigues, M. L. Proença, Anomaly detection using digital signature of network segment with adaptive arima model and paraconsistent logic, in: 2014 IEEE Symposium on Computers and Communications (ISCC), 2014, pp. 1–6. doi:10.1109/ISCC.2014.6912503.

[53] C. E. Shannon, A mathematical theory of communication, The Bell System Technical Journal 27 (3) (1948) 379–423. doi:10.1002/j.1538-7305.1948.tb01338.x.

[54] M. Wang, Y. Lu, J. Qin, A dynamic mlp-based ddos attack detection method using feature selection and feedback, Computers & Security 88 (2020) 101645. doi:https://doi.org/10.1016/j.cose.2019.101645.

[55] P. Dixit, S. Silakari, Deep learning algorithms for cybersecurity applications: A technological and status review, Computer Science Review 39 (2021) 100317. doi:https://doi.org/10.1016/j.cosrev.2020.100317.
URL https://www.sciencedirect.com/science/article/pii/S1574013720304172

[56] M. P. Novaes, L. F. Carvalho, J. Lloret, M. L. Proença, Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment, IEEE Access 8 (2020) 83765–83781. doi:10.1109/ACCESS.2020.2992044.

[57] F. Chollet, et al., Keras, https://keras.io (2015).

[58] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow.org (2015).
URL http://tensorflow.org/

[59] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, Journal of Machine Learning Research 15 (56) (2014) 1929–1958.
URL http://jmlr.org/papers/v15/srivastava14a.html

[60] Y. Wang, J. Liu, J. Mišić, V. B. Mišić, S. Lv, X. Chang, Assessing optimizer impact on dnn model sensitivity to adversarial examples, IEEE Access 7 (2019) 152766–152776. doi:10.1109/ACCESS.2019.2948658.

[61] I. Sharafaldin, A. H. Lashkari, S. Hakak, A. A. Ghorbani, Developing realistic distributed denial of service (ddos) attack dataset and taxonomy, in: 2019 International Carnahan Conference on Security Technology (ICCST), 2019, pp. 1–8. doi:10.1109/CCST.2019.8888419.

[62] M. Team, Mininet overview, online Access: 2020-10-27 http://mininet.org/overview/ (2020).
URL http://mininet.org/overview/

[63] P. Floodlight, Floodlight, online Access: 2020-10-27 http://www.projectfloodlight.org/ (2020).
URL http://www.projectfloodlight.org/

[64] P. Biondi, the Scapy community, Scapy, online Access: 2020-10-27 http://www.secdev.org/projects/scapy/ (2020).
URL http://www.secdev.org/projects/scapy/

[65] M. V. de Assis, L. F. Carvalho, J. J. Rodrigues, J. Lloret, M. L. Proença Jr, Near real-time security system applied to sdn environments in iot networks using convolutional neural network, Computers & Electrical Engineering 86 (2020) 106738, dOI:10.1016/j.compeleceng.2020.106738. doi:https://doi.org/10.1016/j.compeleceng.2020.106738.

[66] R. Priyadarshini, R. K. Barik, A deep learning based intelligent framework to mitigate ddos attack in fog environment, Journal of King Saud University - Computer and Information Sciencesdoi:https://doi.org/10.1016/j.jksuci.2019.04.010.