A Flow-Based Anomaly Detection Approach With Feature Selection Method Against DDoS Attacks in SDNs

Mahmoud Said El Sayed, Nhien-An Le-Khac, *Senior Member, IEEE*, Marianne A. Azer, and Anca D. Jurcut

***Abstract*—Software Defined Networking (SDN) is an emerging network platform, which facilitates centralised network management. The SDN enables the network operators to manage the overall network consistently and holistically, regardless the complexity of infrastructure devices. The promising features of the SDN enhance network security and facilitate the implementation of threat detection systems through software applications using open APIs. However, the emerging technology creates new security concerns and new threats that do not exist in the current traditional networks. Distributed Denial of Service attacks (DDoS) are one of the most rampant attacks that can interrupt the functionality of the network and make most of the network services unreachable for network users. The efficient identification of DDos attacks on SDN environments in literature is still a challenge because of the number of network features taken into account and the overhead of applying machine learning based anomaly detection techniques. Hence, in this paper, we aim to use two popular feature selection methods, i.e., Information Gain (IG) and Random Forest (RF) in order to analyse the most comprehensive relevant features of DDoS attacks in SDN networks. Using the most relevant features will improve the accuracy of the anomaly detection system and reduce the false alarm rates. Moreover, we propose a Deep Learning (DL) technique based on Long Short Term Memory (LSTM) and Autoencoder to tackle the problem of DDoS attacks in SDNs. We perform our analysis and evaluation on three different datasets, i.e., InSDN, CICIDS2017 and CICIDS2018. We also measure the overhead of the proposed DL model on the SDN controller and test the network performance in terms of network throughput and end-to-end latency. The results validate that the DL approach can efficiently identify DDoS attacks in SDN environments without any significant degradation in the controller performance.**

***Index Terms*—Anomaly detection, autoencoder, DDoS, deep learning, LSTM, InSDN dataset, SDN, traditional network.**

I. Introduction

THE TRADITIONAL IP networks, which are widely applied today have become complex and difficult in their management. If the IT operators need to configure any high-level network policies, such as Quality of Service (QoS) or routing policy, they have to access the network devices (e.g., routers and switches) separately using the vendor-specific commands, which increases the overall complexity of the network. Additionally, the IP-based network devices are vertically integrated. The control plane (responsible for the decision-making) and the data plane (which decides how to forward the network traffic according to the instructions from the control plane) are embedded into the same network device. Coupling the control and data planes can hamper the innovation of the network infrastructure and reduce the flexibility of the network for any change or update. Besides, the rapid growth of networking can increase maintenance costs and significantly reduce network innovation in traditional networks. Therefore, developing a new routing algorithm could take 5 to 10 years and would practically be very costly [1]. Moreover, since all devices are widespread through the entire network, there is an increase in the number of middle-boxes devices such as firewalls, load balancers, detection and defense systems, etc. According to Kreutz *et al.* [1], 57 of network enterprises reported that the number of middle-boxes devices has significantly increased and reached the same number of other mandatory network devices like routers.

To address many of the traditional IP network limitations, the emerging network architecture, which is often known as Software Defined Networking (SDN), offers faster failover and enables the network to be centrally controlled. The key idea behind the SDN is to abolish vertical integration by splitting the underlying infrastructure devices from the control plane. The key feature of SDN versus traditional network is shown in Fig. 1. Decoupling the two plane layers increases the network flexibility and facilitates network management with the aid of centralised controller. The new paradigm allows the operators to manage the entire network using software APIs connected with the SDN controller through the northbound interface regardless of the underlying network technology. The global visibility introduced by the SDN system encourages many business enterprises such as Google, Huawei, Microsoft to implement the new paradigm in their network data centre [2].

Despite all benefits offered by SDN, security is one of the significant challenges, which can slow down its widespread adoption and deployment over different networks. Since the centralised controller is the heart of the network, it is vulnerable to a single point of failure. In case the attacker successfully

Manuscript received 8 October 2021; revised 25 February 2022 and 18 April 2022; accepted 19 June 2022. Date of publication 28 June 2022; date of current version 9 December 2022. The Research Funding is University Collage Dublin (UCD). The associate editor coordinating the review of this article and approving it for publication was M. Chen. *(Corresponding author: Mahmoud Said El Sayed.)*

Mahmoud Said El Sayed, Nhien-An Le-Khac, and Anca D. Jurcut are with the School of Computer Science, University College Dublin, Dublin, D04 V1W8 Ireland (e-mail: mahmoud.abdallah@ucdconnect.ie; an.lekhac@ucd.ie; anca.jurcut@ucd.ie).

Marianne A. Azer is with the National Telecommunication Institute, Nile University, Cairo 12677, Egypt (e-mail: mazer@nu.edu.eg).

Digital Object Identifier 10.1109/TCCN.2022.3186331

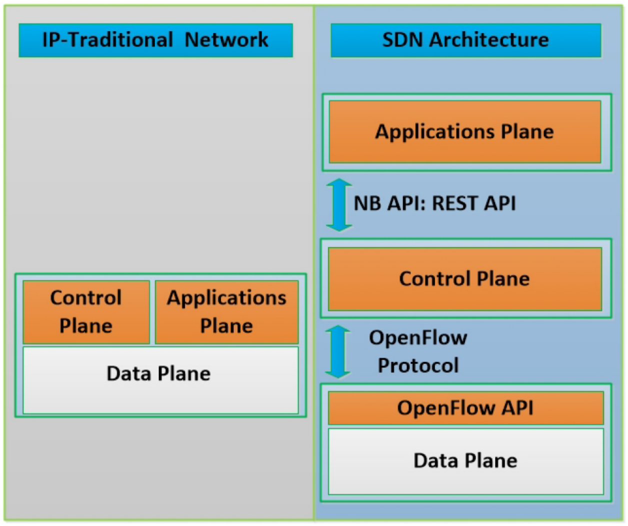


Fig. 1. Traditional vs SDN architecture.

exploits the controller system, he can hinder or manage the entire network based on his aspiration. DDoS is one of the most critical threats in SDN networks. Unfortunately, all SDN layers, i.e., data, control, and application planes are targets to DDoS attacks. Besides, the communication channels between the data plane devices and the control plane have become a potential target for DDoS attacks. Some mitigation techniques, as described in [3] suggested a secondary controller to reduce the damage resulted from DDoS attacks. However, using a secondary controller is not a practical solution to solve the problem since it can also be susceptible to DoS/DDoS attacks.

Intrusion Detection Systems (IDSs) are standard security solutions to monitor and detect malicious activities inside an organisational network. If the observed traffic from the incoming or outgoing network is matched with suspicious activity, an alarm is generated, referring to a detected attack. Therefore, the development of IDSs is a vital direction for many researchers [4], [5] since the security challenges are among the most critical issues facing SDNs. The statistical, Machine Learning (ML) and DL techniques are widely applied for anomaly-based detection[1] solutions [6]. The centralised control plane architecture in SDN provides new opportunities to defeat against DDoS attacks. Motivated by this fact, we apply the DL techniques to temper the problem of DDoS attacks in SDNs.

On the other hand, feature selection methods are one of the significant pre-processing phase to success the anomaly detection models [7]. Such techniques can eradicate the irrelevant and redundant features, retaining only the most representative characteristics from the original dataset. Using optimised subset features not only improves the accuracy and detection rate of the classifier, but also reduces the execution time. So, using less number of features can help to develop a lightweight model able to detect malicious attacks in real time network with low computational resources and prediction latency. In addition, avoiding the curse of dimensionality through the feature selection methods makes the model less prone to overfitting problem. Thus, removing significant noisy and useless features has gained the attention of many researchers to use feature selection strategies in many cybersecurity intelligence solutions to achieve a high model performance using ML/DL tasks [8], [9].

[1]In this article, we use intrusion detection and anomaly-based detection systems interchangeably to refer for the same concept.

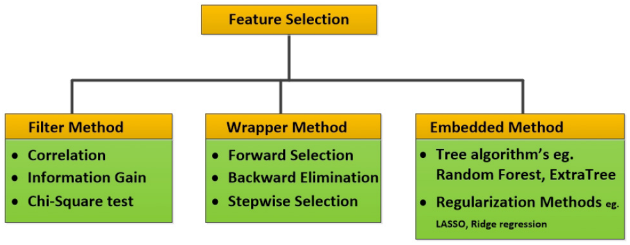


Fig. 2. Methods for Feature Selection.

The feature selection can be categorised into three general methods: filter, wrapper and embedded methods. Some examples of each method is shown in Fig. 2. The interested reader may refer to [10], [11] for more details regarding the different approaches of the feature selection methods.

Although several feature selections with ML models have been proposed to detect DDoS attacks [8], [12], [13], the existing mechanisms to prevent DDoS attacks are ineffective on SDNs. However, one of the significant limitations associated with the aforementioned work is the lack of the intrusion dataset for the SDN network. The researchers widely used a dataset generated based on the conventional network, i.e., not the SDN architecture. However, this adaptation may not be close enough to a real detection techniques in SDNs [14]. SDN has brought its own security threats, and the nature of these threats is different from those commonly affecting legacy networks. For example, all unmatched flows at the open flow switches are triggered to the SDN controller for the policy request. The intruder can send huge amount of unmatched flows to overwhelm the controller resources creating a new kind of DDoS attack. In addition, the attack traffic mimics the same normal behavior since the normal and malicious traffic is forwarded to the SDN controller for decision making. Therefore, the relevant features of DDoS attacks based on conventional networks are not necessarily related to DDoS class on the SDN network. Moreover, using a weak feature selection algorithms will omit the most relevant parameters and this can waste significant data information.

Experiencing with the success of DL in several domain areas, a combination of SDN and DL can improve the performance of intrusion detection systems and then secure the network better. However, as network speed becomes faster, there is an emerging need for IDS to be lightweight with high detection rates. Therefore, feature selection is a significant issue and plays a crucial role in intrusion detection to achieve maximal performance. The efficient feature subset can improve the training and testing time that helps to build lightweight IDS guaranteeing high detection rates and making IDS suitable for real-time and online detection of attacks. In this context, we evaluate the most relevant features of DDoS attacks with the ranked top 10 features obtained by using two common feature selection methods: Information Gain (IG) and RF strategies. Several experiments based on three different datasets have been taken to look at the impact of feature selections on

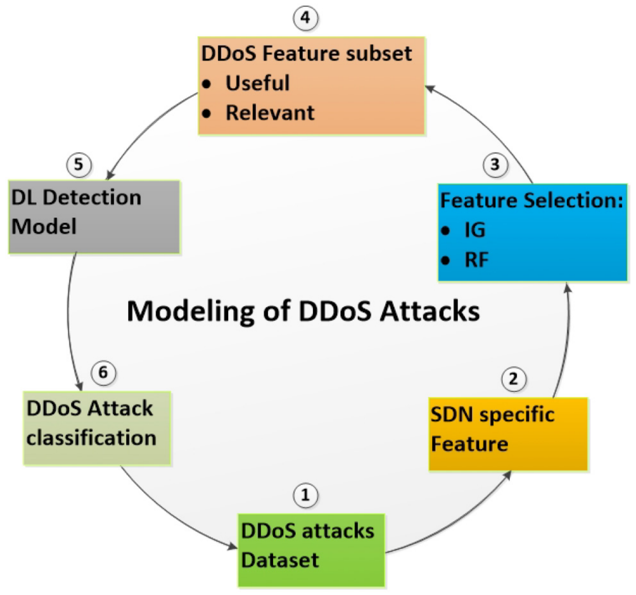


Fig. 3. DDoS Modeling with Feature Selection Process.

the classifier accuracy, execution times. The DDoS modeling process is depicted in Fig. 3.

The key contribution of this work can be summarised as follows:

- Two popular feature selection methods (IG and RF) are used to find the most relevant DDoS attack features in each dataset individually. The proposed feature selection method is tested on three benchmark flow based datasets, i.e., InSDN [14], CICIDS2017 [15] and CICIDS2018 [15].
- A DL based IDS process (Fig. 3) to detect DDoS in SDN. This process includes an extension of our previous DDoSnet model [16] and our feature selection approaches. The results show that using selected feature methods with the proposed system helps in reducing model complexity without any effect on the accuracy of the model.
- Analysis the network performance of the DL model on the SDN controller.The result analysis shows that the DL approach does not significantly degrade the performance of the controller.

The rest of this article is organised as follows: Section II briefly provides a theoretical background about SDN operation and some of the security challenges of the new paradigm. Related work and various detection and defense techniques against DDoS attacks in SDNs are discussed in Section III. Section IV introduces the methodology, the datasets and the DL model used in this work. The experimental and evaluation results are presented in Section V. The network evaluation is discussed in Section VI. Finally, an overall discussion of the results obtained and the conclusion are presented in Sections VII and VIII, respectively.

II. Background Theory

A. SDN Operation

The SDN network comprises three functional planes: the application, control, and data planes [17]. The application plane facilities the deployment of several applications and services through northbound programming interfaces (APIs). For example, applications such as IDS, monitoring, QoS, load-balancer, and many other applications that define the network behavior or offer services for end users can be implemented easily as API. The control plane facilitates the management of the network from a centralised location. The last layer, i.e., data plane layer or underlying network infrastructure contains the forwarding network devices, e.g., OpenFlow switches. The controller is separated from the under-layer devices, and the communication between the two layers is established using Southbound Interface. The OpenFlow protocol has become the *de facto* protocol for communication mechanisms between the controller and underlying switches. For any incoming flow, the switch will search if there is any matching entity in one of its flow tables to handle this flow accordingly. In case of matching, the flow traffic will be directed to the corresponding destination. Otherwise, the switch will extract the packet header, encapsulate it in the format of `Packet-In` message and send it to the controller for further processing. The controller takes decisions on the incoming flows (e.g., flow forwarding or dropping) with the assistance of API programming and returns the flow rule to the switch in the format of `Packet-Out` message. Then, the switch takes the corresponding action according to rules/policies assigned by the controller. The new rules will be cached in the flow table to match any similar flow for a period of time.

B. DDoS Attack Overview

The DDoS attack is an explicit attempt to prevent legitimate users from accessing the network services. The emergence of the DDoS attacks typically does not occur suddenly, but the onset of an attack on a target system produces from a series of preparatory steps by the adversary that we can identify and measure. The operation of DDoS attacks follows several consecutive phases as shown in Fig. 4 [18]. The intruder initially starts to compromise multiple agent machines that are widely distributed geographically by scanning the vulnerabilities in these devices. Once an intruder successfully identifies some system vulnerabilities, he can compromise these machines using a malicious program such as Trojan Horse. By replicating the malicious file in multiple agents, the intruder has the capability to control many devices that can reach several thousand or millions (commonly referred to as bots) to initiate DDoS attacks without the awareness of the device's rightful owner. The discovery of vulnerabilities and exploitation process of the agents are usually performed automatically, for instance, by sending e-mail messages with the attack code attachment. The groups of bots, known as a botnet can get orders remotely from an intruder, i.e., bot-master. The bot-master can perform large-scale DDoS attacks to flood a legitimate service or network by sending a control command to the botnet agents to generate useless traffic without getting noticed. Consequently, the victim resources become overwhelmed with a crushing volume of traffic in a short duration, which significantly slows down the system service or the network ability to respond to the legitimate users.

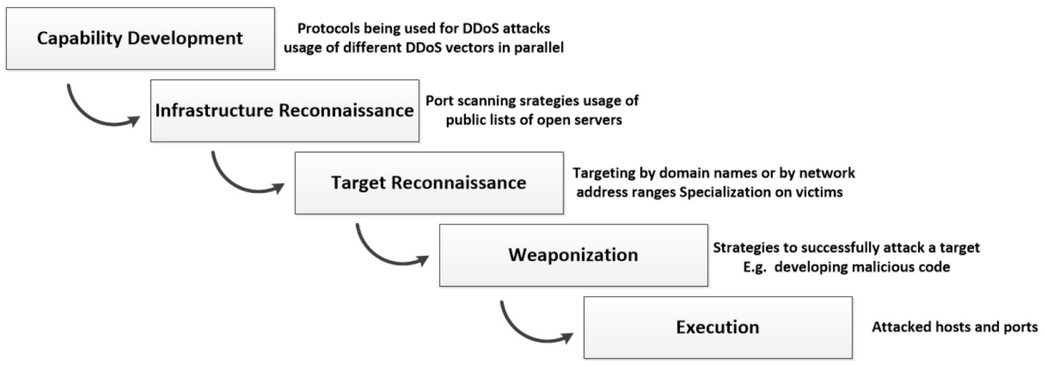

Capability Development
Protocols being used for DDoS attacks usage of different DDoS vectors in parallel
Infrastructure Reconnaissance
Port scanning srategies usage of public lists of open servers
Target Reconnaissance
Targeting by domain names or by network address ranges Specialization on victims
Weaponization
Strategies to successfully attack a target E.g. developing malicious code
Execution
Attacked hosts and ports

Fig. 4. Adversarial Tactics in DDoS Attacks [18].

C. DDoS Attack in SDN

Although there are significant benefits of SDN in several application domains, several security issues in SDNs remain unaddressed. Indeed, the security in SDN is a double edge sword. The centralised location of the controller can improve the overall network security using new security tools with the help of the northbound APIs. However, splitting the control plane from the data plane produces new weaknesses that lead to attacks which did not exist before in the IP-based networks. Examples of these attacks include attacks against the SDN controller or the attacks on the communication links between the controller and underlying infrastructure devices [14], [19]. In addition, all reported attacks in the current networks can also target the SDN network (e.g., application attacks). However, the consequences of the attacks in SDN networks are very significant and can cause crucial damage. On the other hand, the influences of the same attacks in traditional networks are mild or moderate, since only a small part of the network is being affected (likely for the same vendor devices) [19]. When the intruder needs to extend his attacks against new subnets, additional privileges or new attacks are required for these purposes.

One of the most serious attacks in SDN is the DDoS. The attacker can easily generate a high volume of traffic from spoofed IPs, causing heavy damages to the network and making the controller unreachable for the legitimate users. Unfortunately, all SDN layers are susceptible to DDoS attacks [20], and these attacks have different intuition from those reported in the traditional networks, even from the ones that are categorised under the DDoS class. In the following paragraphs, we will emphasise some of the DDoS attacks that are specific for the SDN networks.

- *Buffer Saturation Attacks [21]:* When the switch receives a new packet with no matching entry, the switch extracts the packet header and sends it to the control plane to request a new flow rule. At this time, the packet payload is temporarily buffered in the memory until new instructions are being received from the controller. In case the buffer memory becomes full and has not enough space to store new data, the switch will send the full packet size to the control plane. The attacker can exploit this gap by generating a vast number of fake packets with forged IP addresses to run out the buffer memory within a short time. When there is no buffer space, the legitimate packets are unable to buffer too, resulting in buffer saturation attacks.
- *Flow Table Overflow [22]:* The switch flow tables are stored in a memory, known as Ternary Content Addressable Memory (TCAM). Each entity rule associated with it is defined with two times, i.e., idle timeout and hard timeout to address the limited space of OpenFlow switches. The idle timeout is referred to the amount of time in seconds when the flow is removed from the flow tables in case no flow is matching it. The hard timeout determines how long this flow will stay in the flow table before being removed, whether or not the flows match it. The attacker can use this feature and send a large number of the unmatched flow. After a while, all flow entities will be replaced by fake flows and the memory gets full with useless rules. Simultaneously, the switch will fail to handle any legitimate users and all received flow will be dropped. However, the switch can handle a limited number of incoming packets, since the TCAM memory has limited space. This is because the TCAM cost can reach 400 times or over the RAM cost and its usage power reaches 100 times that RAM consumes [23].
- *Link Flooding Attack (LFA) [24]:* The flow switches communicate with the SDN controller using southbound links. In case the intruder generates numerous fake packets and no buffer space in the switches, the full packet will be delivered to the SDN controller, and this can quickly overload the bandwidth, creating a bottleneck for the legitimate traffic.
- *Controller Saturation [25]:* The controller is an application installed on a virtual machine and has limited resources, such as RAM and processing power. When the controller handles a large number of fake packets, the extensive processing can degrade its resources. The controller saturation attack has a critical affect on the SDN controller, since any breakdown or failure for the controller causes that the entire network to be lost.

The SDN controller is highly targeted by DDoS attacks and can quickly become a bottleneck if it handles a large amount of incoming flow. Since all unmatched packets are relayed to the controller for drawing the new rules, then receiving a high number of flows can run out its resources very

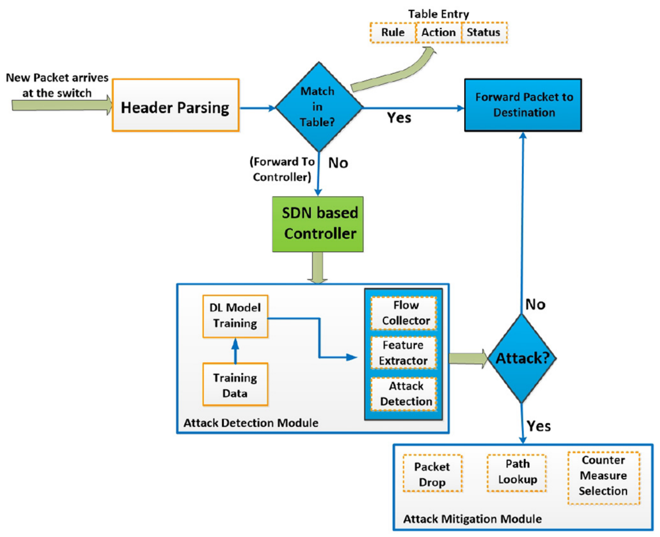


Fig. 5. Anomaly detection and mitigation Framework in SDN.

fast, regardless the effectiveness of the controller. The overwhelming of the controller can cause the utmost damages and may crash the entire SDN network. Therefore, protecting the network resources from DDoS attacks has become a crucial aspect for the researchers in recent years.

D. Flow-Based IDS in SDN

The effective detection approaches are running on top of the controller as a REST API. The centralised behavior of the SDN architecture allows the controller to take the corresponding action according to detecting results. Figure 5 represents the SDN framework to discriminate benign traffic from ordinary DDoS attacks. The architecture of the framework consists of three major modules as follows:

- *Flow Collection and Extractor Module:* This unit gathers flow statistics from the incoming packets using query rules initiated by the controller every periodic interval. The controller requests the switch to send the flow statistics for analysis through a standard `OFPT_FLOW_MOD` message. The time interval plays an important role on the efficiency of the detecting module. If the time interval is very long, it can raise the workload on the controller and switches, since a huge number of flows are processed. Besides, the detection module can take a long time to respond, giving the attacker a good chance to harm the network with aggressive fake traffic without being detected. In contrast, when the trigger time is very short, the controller will initiate the detection module very fast, and this can increase the computational cost and increase the controller resource occupation. Additionally, a high volume of traffic will be transferred between the controller and switches, and this may consume the links bandwidth in a short time. To solve the aforementioned problem, few works were conducted [26], [27] to improve the mechanism for the flow data collection, to avoid the overhead and high computational cost on the control plane. The controller will extract the specific characteristics from the collected data. The extracted features are essential to discriminate between normal and DDoS malicious traffic.
- *Identification Module:* This is the core of the framework and comprises the trained detection module, e.g., IDS. This module performs the identification on the attributes extracted from the previous step in order to distinguish whether the incoming flow is malicious or normal. The IDSs are widely classified into two different classes: signature-based and anomaly-based [28]. The signature-based systems (e.g., Snort) match the signature of attacks with some rules stored in an acknowledge database. Such techniques achieve high accuracy (i.e., can reach over 99%), but unfortunately, their performance is abysmal in detecting zero-day attacks. The attacker can easily bypass the functionality of signature-based techniques without being notified if s/he successfully manipulates the attack signature (even for small amendments). On the other side, anomaly-based techniques have received a significant attention from the research community in the last decade, since they theoretically have the capability to detect new attacks by observing any deviation from the normal traffic pattern. However, the anomaly-based detection solutions suffer from high false alarms, which can slow down their implementation on network productions or commercial products. The quality of any anomaly detection system relies on the quality of the training dataset. Therefore, the dataset should be updated periodically in order to include the new attack patterns. The DDoS detection module will analyse each received packet before the controller processes it. If the received packet is normal, the controller will instruct the switch to install a new flow rule, while the malicious traffic will be sent to the next mitigation management module to handle it.
- *Mitigation Management Module:* Whenever a malicious flow is identified by the identification module, the controller utilises the corresponding attack defense measures as soon as possible to avoid any damage to the network. Various mitigation strategies are broadly applied to deal with the incoming malicious traffic. The most popular solution is to block the attack flows by activating a new flow entry in SDN switches with the action field set as Drop. Another mitigation solution, introduced by Alshamrani *et al.* [29] is to move all the excessive malicious flows to another honeypot server for further detection as shown in Fig. 6 [29] shows that the SDN controller receives a large amount of traffic packets at the beginning of DDoS attacks, while the amount of received packets at the honeypot is significantly low. Once the detection module identifies the attacks, all traffic is redirected to the honeypot server for further investigation. Thus, the amount of received packets at the controller decreases with time, while the honeypot receives a large amount of traffic size. Therefore, the high false alarms of the detection module can be avoided and its performance in unknown attacks will be enhanced. Additionally, it is also important to remove the malicious flow entries from

TABLE I
SUMMARY OF THE STATE-OF-ART SOLUTIONS AGAINST DDoS ATTACKS IN SDNs

Type	Publication	Description Of the Solution	Used Dataset	Advantages	Limitation
Statistical Information Entropy Based Solutions	Prashant*et al.* [31]	An entropy-based detection mechanism was used to mitigate TCP SYN flooding attacks in the SDN. TCP flags and destination IP were used to calculate the value of the entropy.	-	- Procide mechanisms, which are lightweight, fast in their calculation and consume less amount of computational resources	- Broadly depend on the security researchers' experience to define beforehand the threshold value. However, the network traffic is dynamic and is not consistent all the time. - Selecting the optimal threshold value need several claculations and requires the past observation of the network behaviour - The modern attack traffic has a high degree of similarity with normal traffic e.g. Low-Rate DDoS attack), which is hard to be identified using the threshold-based method.
	Kalkan *et al.* [32]	A joint entropy was used for DDoS detectionnthe IP and TCP attribute flags are used to find the value of the entropy.	-		
	Yu *et al.* [33]	combined the entropy scheme with the ensemble learning techniques for DDoS attacks detection. The information entropy was claculated for the destination IP at the edge switches .	-		
	Mishra *et al.* [34]	Shannon entropy was used to recognise and mitigate DDoS attacks. Three thresholds were employed to reduce the false alarm rate.	-		
	Sahoo *et al.* [35]	Generalized Entropy (GE) based metric was used to detect the. low rate DDoS attack to the control layer.	-		
Machine Learning Based Solutions	Tan *et al.* [27]	A detection and mittigation framework agains DDoS attacks. The detection model combined the K-Means and KNN algorithms together.	Simulated		
	Dong *et al.* [39]	Introduced an improved KNN based model for DDoS detection in SDNs.	Simulated		- They have a low ability in the network flow traffic.
	Lingfeng *et al.* [40]	Introduced a detection and mitigation framework for DDoS attacks and performed traffic analysis based on SVM algorithm.	KDD'99 dataset	- Require small amount of data for training	- Sustained with high false alarms and low detection rates.
	Yu *et al.* [41]	Designed an efficient platform for DDoS attacks detection by adopting SVM classification algorithm.	Simulated	- Require less training time	- Achieve reasonable results when the dataset size has low amount of samples.
	JESÚS *et al.* [42]	Applied six ML algorithms, i.e. REP Tree, SVM, MLP, RF, J48 and Random Tree for DDoS attack detection under the SDN context.	CIC-DDoS2019	- Limited tuning capabilities	- Fail to provide significant results when applied on large traffic data.
	Abdulrahman *et al.* [43]	Applied four ML algorithms i.e. C4.5, NB, SVM and RF to solve the problem of DDoS attacks.	CICIDS2017		- Need lot of domain expertise, human intervention
	Bindra *et al.* [44]	Six ML techniques were used in the classification phase to test each method separately. used different feature selection methods to find the most relevant features of DDos attacks.	CICIDS2017		
Deep Learning Based Solutions	Ahuja *et al.* [45]	compared the performance of ANN with various classical ML algorithms for DDoS attacks detection in SDNs.	Simulated	- Significantly solve the inherent problems of traditional ML techniques. - Eliminates the need of domain expertise and hard core feature engineering.	- The existing studies validated their models using a dataset produced based on traditional IP networks and not on SDN platforms. - The SDN encounters new vulnerabilities that can motivate the attacker to easily create new attacks, causing confusion for IDSs in the SDN. - Most of these solutions included few types of DDoS attacks for evelaution process, without considering the attacks that can target all layers. - Moreover, the generated attacks in the simulated datasets were produced using simple tools eg. Scapy or Hping3, and targeted only the network layer of the OSI model, without including the attacks against the application layer.
	tang *et al.* [46]	GRU was introduced to solve the problem of DDoS attacks in SDN networks.	NSL-KDD		
	Li *et al* [47]	Three DL algorithms i.e. CNN, LSTM and RNN were used to built an efficient security defense mechanism using DL algorithms.	ISCX dataset		
	Haider *et al* [48]	Employed four ensemble DL approaches against DDoS attacks in the SDN network.	CICIDS2017		
	Novaes *et al.* [49]	GAN framework to alleviate the impact of DDoS attacks in SDNs.	CIC-DDoS2019		

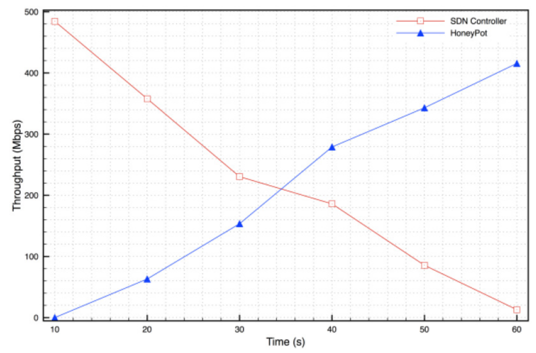


Fig. 6. Traffic burst at SDN controller vs Honeypot [29].

the switches to release the memory space and to avoid any latency process for normal traffic. In practice, the controller sends `OFPFC_DELETE` message to open flow switches to delete the flow entries that consume large storage space resources.

III. RELATED WORK

In recent years, several defense and mitigation techniques have been proposed to tackle the problem of DDoS attacks in SDNs. Although the centralised controlling point of SDN can combat the problem and facilitate the detection of attacks, it can also create new attack vectors that are being reported in traditional networks. In our previous work [30], we demonstrated that the intruder could quickly degrade the network performance or deplete its resources by generating a large number of flows toward the controller with the help of only few hosts under his/her control. Thus, securing the SDN networks from DDoS attacks is essential to keep the network running and avoid any crash to the network services or equipments.

In this section, we explore the most comprehensive solution to debate the DDoS attacks under the context of SDNs. Several techniques have been proposed to tackle this problem; either (1) implemented statistical information entropy, (2) machine learning algorithms, and (3) deep learning based solutions. The summary of these different techniques is depicted in Table I.

A. Statistical Information Entropy Based Solutions

Kumar *et al.* [31] introduced a `SAFETY` scheme to mitigate `TCP SYN` flooding attacks in the SDN by using an entropy-based detection mechanism. Only few attributes of `TCP` flags and destination IP were used to calculate the value of the entropy. The proposed technique avoided the static threshold by employing an adaptive threshold during the detection process. However, the proposed approach was dedicated only for `TCP SYN` attack, without considering various types of DDoS that can occur in the network.

Kalkan *et al.* [32] utilised a joint entropy for DDoS detection under the SDN context. Similar to [31], the IP and `TCP` attribute flags are used to find the value of the entropy. However, the utilised method provided acceptable results only for known attacks and failed to provide a desirable accuracy for any types of unfamiliar attacks.

Yu *et al.* [33] combined the entropy scheme with the ensemble learning techniques for DDoS attacks detection in SDNs. A lightweight model based on entropy strategy was adopted at the edge switches to calculate the information entropy of destination IP. In case of suspected anomalies, another detection technique based on Random Forest (RF) will be initiated on the controller for further detection. However, the authors employed the open flow switches for detection tasks, which is against the core functionality of SDN, to avoid any decision making in the underlying forwarding devices.

Mishra *et al.* [34] used Shannon entropy to recognise and mitigate DDoS attacks in SDN networks. Three thresholds were employed to reduce the false alarm rate. The authors claimed that the proposed method achieved an accuracy of over 98.2% with a false-positive rate of 0.04%.

Sahoo *et al.* [35] proposed a Generalized Entropy (GE) based metric to detect the low rate DDoS attack on the control layer. The experimental results showed that the Generalized Entropy achieved better performance compared to Shannon metric and with other information distance metrics.

Although all mechanisms based on entropy are lightweight [36], fast in their calculation and consume less amount of computational resources, these techniques broadly depend on the security researchers' experience

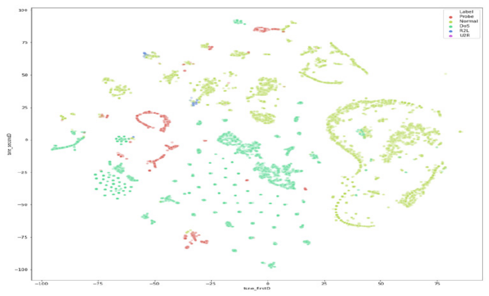


Fig. 7. Visualisation result of NSl-KDD dataset. The t-SNE algorithm is used to understand the distribution of data intuitively.

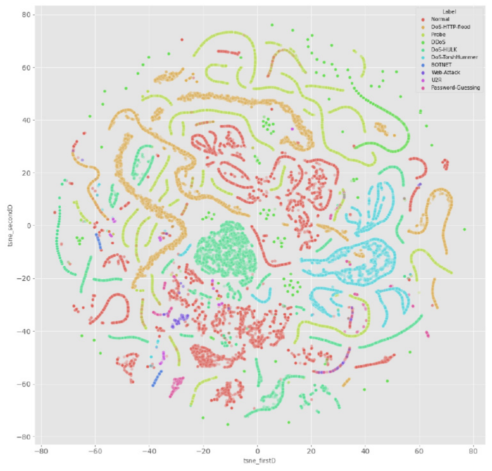


Fig. 8. Visualisation result of InSDN dataset.

to define beforehand the threshold value. However, the network traffic is dynamic and is not consistent all the time. For example, the size of traffic is relativity high during the production time, while it becomes less in the evening or on weekend days. Therefore, selecting the optimal threshold value needs several calculations and requires the past observation of network behaviour. Also, the nature of network traffic is changed continuously over time, i.e., new protocols are developed frequently, while other protocols are no longer used. Consequently, the modern network traffic is very complex since the intruder can easily create new attack traffic with a high degree of similarity with normal traffic (e.g., Low-Rate DDoS attack), which is hard to be identified using the threshold-based method. We discussed this issue in our previous research work [37], [38]. Selecting a sub-optimal threshold value is reliable for old network data, which was produced long time ago since the cluster of normal and attack traffic are spatially separated from each other, as shown in Fig. 7. While the clusters of modern traffic data are spatially combined (Figure 8), tuning the threshold substantially leads to higher false-alarm instances, since the mitigation module is not able to discriminate between normal and malicious instances. Hence, threshold-based methods are not significantly conceived for detection and classification applications. Therefore, valid and reliable security solutions are still needed to efficiently secure the networks from malicious traffic.

B. Machine Learning Based Solutions

In recent days, ML based anomaly detection techniques are efficiently used for DDoS attacks detection in SDN networks. The ML has the capability to learn and identify patterns from data automatically with the help of the training data. These techniques have the capability to detect the abnormal behaviour of the network data and provide a better performance than the signature based techniques.

Tan *et al.* [27] introduced a novel detection and mitigation framework to defend DDoS attacks in SDNs. The framework combined the K-Means and KNN algorithms together for a detection mechanism. 5-tuples entries were employed to identify the DDoS attacks. A new trigger mechanism was developed to reduce the controller workload and to overcome the computational overhead on the communication channels by avoiding periodic traffic collection. However, the authors used the `Scapy` tool to simulate the legitimate and DDoS attacks for creating the training dataset. The simulated traffic lacked the attack variety and was free from any application DDoS attacks. Nonetheless, the application attacks have a high degree of similarity with normal traffic and are not easy to detect using the traditional ML algorithms. In addition, the performance of the detection model was further evaluated on the outdated NSL-KDD dataset, which was generated based on traffic traces from two decades ago.

Dong and Sarem [39] introduced an improved KNN based model for DDoS detection in SDNs. Few attributes (i.e., flow rate, flow size, flow duration, flow length) were used for model training. The dataset was generated using SDN topology, contained one server and ten virtual machines. However, the simulated traces lacked the diversity of DDoS attacks that can occur on different layers of OSI model, such as application layer attacks.

Yang and Zhao [40] introduced a detection and mitigation framework for DDoS attacks and performed traffic analysis based on SVM algorithm. The detection model was evaluated on KDD'99 dataset using only eight features that are easy to obtain from the SDN network. However, the KDD'99 dataset is outdated since it was generated two decades ago and is lacking modern traffic data. In addition, it has a high number of redundancy records, and this can increase the likelihood of overfitting problem.

Yu *et al.* [41] designed an efficient platform for DDoS attacks detection by adopting SVM classification algorithm. The authors utilised the rate of `PACKET_IN` message per time as a trigger mechanism to initiate the classifier model in the SDN controller. The normal traffic in the dataset is based on traces of real traffic collected between Japan and the United States. The DDoS attacks traffic was simulated using `Scapy` and `Hping3` tools. Eight features were extracted for the attack detection process. However, the number of records that are used for training and testing the data are significantly small (i.e., 1200 samples for training and 1700 for testing). Using only few amounts of samples can lead to underfitting problem since the less number of samples are not enough for the model to extract the discriminatory information from the input data.

Pérez-Díaz *et al.* [42] applied six ML algorithms, REP Tree, SVM, MLP, RF, J48 and Random Tree for DDoS attack detection under the SDN context. The CIC-DDoS2019 dataset, was used to validate the performance of proposed approaches. Once the attacks are detected by one of the used ML algorithms, a mitigation strategy is started to block the malicious traffic before crashing the entire network. The IDS module was installed on a separate platform and connected to the controller through an Identification API. Although the implementation of the IDS on an individual platform can reduce the controller workload, it also increases the required resources, which can increase the total cost of the IDS framework.

Abdulrahman and Ibrahem [43] applied four ML algorithms, i.e., C4.5, NB, SVM and RF to solve the problem of DDoS attacks. The IG feature selection method was used in the first stage to select the best 10 features of the CICIDS2017 dataset.

Bindra and Sood [44] used different feature selection methods, i.e., Recursive Feature Elimination (RFE), 'SelectPercentile', 'SelectFromModel', and Principal Component Analysis (PCA) to find the most relevant features of DDos attacks using CICIDS2017 dataset. The size of best-selected features is in the range of 12 to 15 in most feature selection methods. Six ML techniques were used in the classification phase to test each method separately. The RF and KNN provided the highest performance, while LR and NB have the lowest accuracy.

The aforementioned approaches [27], [39]–[44] are based on traditional ML techniques and categorised under shallow learning algorithms. Although these methods are often used and successfully achieve high performance in various applications and domain areas, they have a low ability in the network flow traffic. The shallow learning techniques are sustained with high false alarms and low detection rates, since they cannot meet the requirements to detect complex malicious attacks. On the other hand, these approaches achieve reasonable results when the dataset size has low amount of samples. On the contrary, they fail to provide significant results when applied on large traffic data.

C. Deep Learning Based Solutions

Nowadays, Deep Learning (DL) approaches play a vital role in anomaly detection techniques. Such techniques have the capability to capture the deep structure from the input data automatically without any human intervention. However, only few works utilised the DL for DDoS attacks in SDN networks.

Ahuja *et al.* [45] compared the performance of Artificial Neural Network (ANN) with various classical ML algorithms for DDoS attacks detection in SDNs. A DDoS dataset was created in an emulated environment with the help of mininet and Ryu controller. The results demonstrated the potential of ANN for attack detection with an accuracy that reached 98.2%.

A DL approach [46] based on Gated Recurrent Unit (GRU) was introduced to solve the problem of DDoS attacks in SDN networks. Only six selected features from the NSL-KDD dataset have been employed for DDoS attack classification. The authors claimed that their proposed model achieved an accuracy reached 89% .

Li *et al.* [47] built an efficient security defense mechanism using DL algorithms against DDoS attacks in SDN networks. Three DL algorithms, i.e., CNN, LSTM and RNN were used for the proposed model and the proposed model was evaluated on the ISCX dataset. Their model successfully achieved an accuracy of 99%, and 98% in training and test data, respectively.

Another study [48] employed four ensemble DL approaches against DDoS attacks in the SDN network. The results showed that the deep convolutional neural network (CNN) based model achieved the highest accuracy of 99.45% compared to other hybrid state-of-the-art algorithms. The CICIDS2017 dataset was used to evaluate all proposed DL models.

Novaes *et al.* [49] used Generative Adversarial Network (GAN) framework to alleviate the impact of DDoS attacks in SDNs. The emulated and the public dataset, i.e., CIC-DDoS2019 were used for experiments evaluations. The authors compared the obtained results from the GAN framework with different DL algorithms, e.g., LSTM, CNN, MLP.

Although the DL techniques can significantly solve the inherent problems of traditional ML techniques, most of the existing studies validated their models using a dataset produced based on traditional IP networks and not on SDN platforms. However, the characteristics and the operation behaviour of SDNs are largely different from the current networks. Besides, the SDN uses new protocols (e.g., OpenFlow) that are different from those used in traditional networks. The OpenFlow protocol encounters new vulnerabilities, and this can motivate the attacker to easily create new attacks, causing confusion for IDSs in the SDN. Adding to these factors, many studies are still using outdated datasets, such as KDDCup-'99' and NSL-KDD. These datasets are not only produced based on traces of two decades ago, but they also lack the current Internet traffic. However, the modern intrusion attack types are constantly growing and are becoming more sophisticated, i.e., not easy to identify. On the other side, the previous studies, which emulated the SDN network to create a new dataset for evaluation process, only included few types of DDoS attacks, without considering the attacks that can target all layers. Moreover, the generated attacks were produced using simple tools, e.g., *Scapy* or *Hping3*, and targeted only the network layer of the *OSI* model, without including the attacks against the application layer. However, the DDoS attacks against the application layer are not easy to detect since they are very similar to normal traffic. On the contrary, the DDoS attacks at the network layer are largely deviated from normal traffic and are easy to be detected using simple algorithms.

IV. Methodology

This section discusses in detail our experimental setup, the datasets used for our experiment evaluation, the feature selection methods, and the DDoS detection approach. We explore the potential of DL techniques for DDoS attack detection in the SDN environments. The detailed process of the proposed framework is summarised in Fig. 9. At the first stage, the SDN-specific features are selected manually from three input

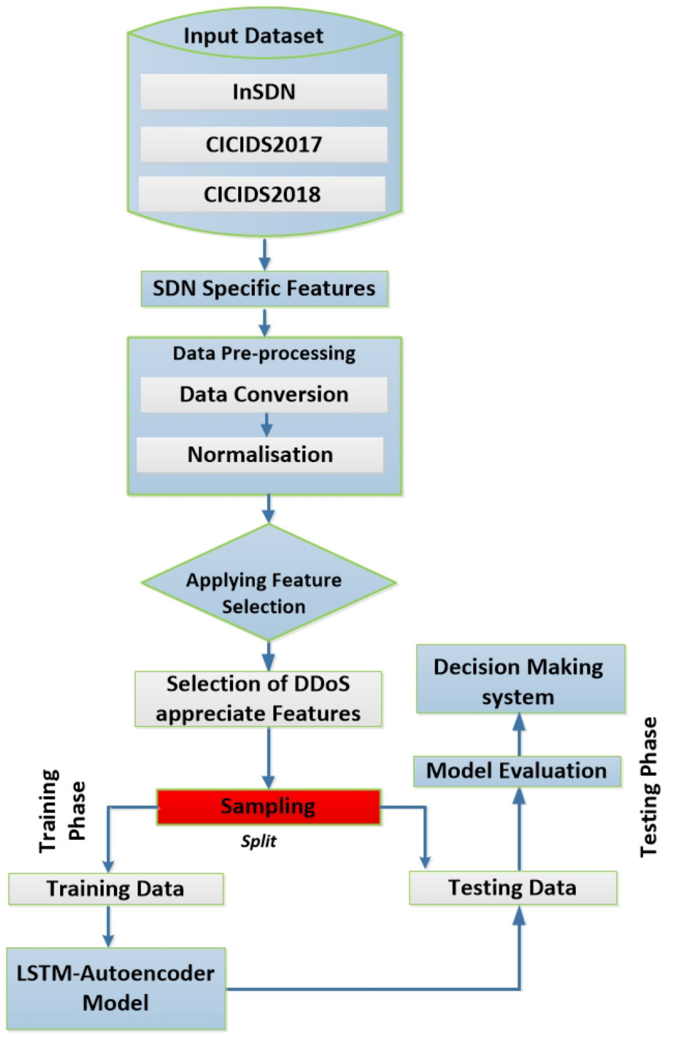


Fig. 9. The flow diagram of the DDoS Detection module.

datasets. Then, various preprocessing steps, as described in Section IV-C are used to fit the input data for the DL model. Additionally, We apply two feature selection methods to find the most relevant features for the DDoS attacks in each dataset. In this work, several experiments are executed to validate the capability of the DL approach for DDoS attacks detection.

A. Dataset Description

The quality of the training datasets plays an essential role in building an efficient anomaly detection-based IDS. However, the availability of high quality datasets for intrusion detection and network traffic, in general, is a significant problem. In different application domains, such as language translation and computer vision, a bunch of various datasets with high quality are available for the public online. On the contrary, the network data can contain sensitive customer information, and it is illegal or against privacy to reveal such data to the public. Hence, we can find the most real datasets for intrusion detection are anonymised payload data, which largely alter the performance of the classifier models [50]. In addition, the majority of the available datasets are outdated, lack traffic diversity, and are unreliable for modern attack detection techniques. Using non-compatible datasets can create a rigid model and may cause a mismatch between the model and the new technology, since the network traffic is dynamic and the enterprises can change the used protocols continuously. For example, the Flash and Silverlight protocols were widely used until 2010 for the most popular video enterprises, i.e., YouTube and Netflix. Currently, they have been replaced with the HTML5 protocol [51]. Therefore, several research works have been proposed to simulate new datasets for the research purpose. The Canadian Institute for Cybersecurity (UNB) is one of the significant centers over the world, that it has contributed to generate reliable and validated datasets. The UNB created intrinsic and publicly available datasets using network typologies that mimic the real network datacenters. Although the produced datasets by UNB are widely used in many research works in SDNs, they were created based on conventional or traditional IP networks, i.e., not from SDNs. However, as previously discussed in Section II, the IP-traditional network and SDN are significantly different in their operation. In addition, decoupling the data plane from the control plane makes the network susceptible to new attack vectors, different from those reported in traditional IP networks. For example, decoupling the SDN controller from the network devices increases the attacker chances to carry out various types of attacks in data communications systems or on the SDN controller itself. However, such attacks are hard to be detected since the attacker is connected to the victim server in an authorised way. Thus, using a non-suitable dataset can mislead the detection system and create high false alarms. In addition to the aforementioned problem, to our best knowledge, there is no publicly available dataset for testing and evaluation of IDS in SDNs environment. The majority of anomaly detection work in SDNs has implemented standard datasets generated based on the conventional network. To tackle all of these problems, we used our InSDN dataset [14] to test the performance of the proposed DL models. In addition, the CICIDS2017 and CICIDS2018 datasets are also used in this work for further evaluation. The description of the three different datasets is discussed, as follows:

- *InSDN [14]:* The InSDN dataset considers the new structure of the SDN network. It was created using four virtual Machines (VMs). One VM acted as an SDN controller, i.e., `ONOS`, while the second VM was used to act as an Open Virtual switch (OVS). The third VM, i.e., Kali Linux was used to represent the intruder machine, while several venerable applications (i.e., Metasploitable2) were installed on the last VM. Additionally, four internal virtual hosts were created using a mininet emulator tool to represent the normal users and some inside malicious hosts. Therefore, the dataset simulated a variety of attack classes from inside and outside the SDN network. The normal traffic in the InSDN dataset reflected several application services, such as `HTTPS`, `DNS`, `SSH`, `FTP`, `email`, etc. For this purpose, some internal hosts are allowed to access the Internet and collect intrinsic traffic from different websites, such as `YouTube`, `Facebook`, `SKYPE`, etc., to mimic the real-world traffic. The total number of instances in InSDN dataset is 361,317, where the size of samples for normal and attack classes is 68,424 and 292,893, respectively.

- *CICIDS2017 [15]:* The dataset contained network traffic of five days, generated in the period between Monday, July 3, and Friday, July 7, 2017. The CICIDS2017 was created using a complete network topology with several devices such as routers, switches, firewalls, and different operating systems platforms. The authors used the concept of profiles to create the normal traffic in the datasets. The dataset was publicly available online in both *PCAP* and *.CSV* formats. The CICIDS2017 includes a total number of instances equal to 2,830,743, where the size of attacks represented 19.7% of the total data.
- *CICIDS2018 [15]:* The authors of [15] extended the CICIDS2017 project to create a new realistic dataset in a scalable manner. The CICIDS2018 traces were gathered in 10 days with a total number of instances 16,233,002, where the size of attacks represented 17% of the entire data. The same concept of profiles was used to create the normal and attack classes, but the authors used the Amazon Web Services (AWS) platform instead of the old network infrastructure.

The three dataset features are generated using the CICFlowMeter tool [52] and have more than 80 network flow features in the format of *.CSV* file. The three datasets contain a variety of attack classes. This work only focuses on DDoS attacks, so the other attack classes are excluded from our study. However, the size of the InSDN dataset is significantly small compared to other datasets, so we take all labels categorised under normal and DDoS classes. Nonetheless, only the Friday afternoon (July 7) file is picked from CICIDS2017 for our experiments, while Wednesday (February 21) file is used in the case of the CICIDS2018 dataset.

B. SDN Specific Features

This section explains some representative features that can be derived in SDN networks. The features of the three datasets were obtained using the open source CICFlowMeter tool [52]. The CICFlowMeter generates more than 84 flow features. However, not all of these features can be extracted inside the SDN environment. In SDN, only statistical features can be extracted from the SDN controller through OpenFlow calls to the SDN switches (e.g., flow duration, number of packets, number of bytes). For this goal, we use the same framework of [53] to find the sub-features, which are easily retrieved directly by the SDN controller quarries or by competition calculation of the flow statistics. For example, we can use manual computational to calculate some features such as standard deviation (Std), Min, Max, and mean of the flow features. Table II represents the corresponding mapping between derived features from the SDN environment to the InSDN dataset features. In addition, Table III shows extra features that can be calculated from the manual competition. Nonetheless, the original framework [53] utilised a subset of 50 features for their research objective. In this article, a subset of 48 features is only used, as the source and destination IPs are excluded from our experiments. The two attributes may be changed from one network to another; besides, the attacker can use the same IP address of legitimate users. Thus, training the classifier model using such features can make the model biased toward those socket features, causing the overfitting problem. The obtained features in SDNs are depicted in Table IV.

TABLE II
THE EXTRACTED TRAFFIC FEATURES FROM SDN CONTROLLER [53]

No.	Feature Description	SDN Derived Features	InSDN Dataset
1	Length of the connection	Duration	Flow Duration
2	Protocol_type	Protocol	Protocol
3	Max. expire time of flow	Hard_time_out	Flow use
4	Flow permanence time	Idle_time_out	Flow idle
5	Packets in bidirectional flow	Packets	Packets
6	Data bytes in bidirectional flow	Bytes_count	Bytes
7	Data bytes from source to dest.	Tx_packets	Src2dst_packets
8	Data bytes from dest. to source	Rx_packets	dst2src_packets

TABLE III
THE EXTRA TRAFFIC FEATURES [53]

No.	New Feature	InSDN Feature
1	Packet/s from source to dest. (PPS)	Packet rate (src2dst)
2	Packet/s from dest. to source	Packet rate (dst2src)
3	Inter-arrival time (IAT) (min, avg, max, std)	Inter time
4	Inter-arrival time from source to dest.(min, max, mean, std)	Inter time (src2dst)
5	Inter-arrival time from dest. to source (min, max, mean, std)	Inter time (dst2src)

TABLE IV
THE 48 EXTRACTED SUBSET FEATURES IN SDNS

No.	Feature Name	No.	Feature Name
1	Protocol	25	Forward_IAT_Total
2	Flow_duration	26	Backward_IAT_Min
3	Total_Length_of_Fwd_Packets	27	Backward_IAT_Max
4	Total_Length_of_Bwd_Packets	28	Backward_IAT_Std
5	Total_Forward_Packets	29	Backward_IAT_Mean
6	Total_Backward_Packets	30	Backward_IAT_Total
7	Forward_Packet_Length_Std	31	Forward_Header_Length
8	Forward_Packet_Length_Mean	32	Backward_Header_Length
9	Forward_Packet_Length_Min	33	Forward_Packet_s
10	Forward_Packet_Length_Max	34	Backward_Packet_s
11	Backward_Packet_Length_Std	35	Max_Packet_Length
12	Backward_Packet_Length_Mean	36	Min_Packet_Length
13	Backward_Packet_Length_Min	37	Packet_Length_Mean
14	Backward_Packet_Length_Max	38	Packet_Length_Std
15	Flow_Packet_s	39	Packet_Length_Variance
16	Flow_Bytes_s	40	Average_Packet_Size
17	Flow_IAT_Min	41	Active_Min
18	Flow_IAT_Max	41	Active_Max
19	Flow_IAT_Std	43	Active_Std
20	Flow_IAT_Mean	44	Active_Mean
21	Forward_IAT_Min	45	Idle_Min
22	Forward_IAT_Max	46	Idle_Max
23	Forward_IAT_Std	47	Idle_Std
24	Forward_IAT_Mean	48	Idle_Mean

C. Data Preparation

Data preprocessing is a crucial step taken on the input data before the model training to build an accurate detection system. The original data is not suitable for building and training ML/DL models; hence some steps are taken to transform the input dataset into an understandable and readable format as follow:

TABLE V
THE SIZE OF SAMPLES IN DATASETS

Dataset	training		testing	
	Normal	DDoS	Normal	DDoS
InSDN	47,732	85,524	20,692	36,418
CICIDS2017	68,436	89,561	29,250	38,464
CICIDS2018	252,673	240,619	108,160	103,252

- The CICIDS2017 and CICIDS2018 datasets contain a huge amount of infinity and missing (nan) values. The first step to build an accurate model is to clean the data. In practice, two different methods can be taken to handle the missing and infinity values in any particular column, which has the missing values. We can either remove these values or calculate the mean and replace them with the results. In this article, since the two datasets have adequate samples, we drop all null and infinity values without causing any significant effect on the model efficiency.
- ML/DL techniques are based on mathematical equations, so the categorical data are converted into numerical values to keep only numbers in the equations. The `OneHotEncoder` class is used to replace the text of the labeled column with number. In these experiments, only binary classification is employed to classify normal or DDoS attacks. Thus, the normal traffic takes the value of 0 and DDoS attacks are encoded to the value of 1.
- The dataset features have different scales, and this can cause some issues in the DL model. For example, some dataset columns or features have a small range of values, while other columns take a large range of values, i.e., higher than the value in another column. We limit the range of variables by using `feature scaling`, so the common ground can be used for the comparison. There are two different methods of scaling: `normalization` and `standardisation`. The `normalization` scales the features between 0 and 1, while the `standardisation` converts the input attributes into a new scale, which has a zero mean (μ) and a standard deviation (σ) of 1. In this work, we applied the `standardisation` method for all datasets according to Eq. (1).

$$x(i) = \frac{x(i) - \mu(x(i))}{\sigma(x(i))}. \quad (1)$$

- We split the dataset into a 70:30 ratio using `test_train_split` from the `sklearn` library, which means that 70% of the dataset are utilised for training, while the remaining 30% are reserved for the model test to check how accurately we can predict it. The total number of training and testing samples for all datasets is depicted in Table V.

D. Deep Learning Classifier

Recently, DL techniques have gained popularity on a broad variety of tasks, e.g., speech recognition, computer vision, language translation. DL techniques use multiple hidden layers to solve more complex problems, which are difficult for solving using a linear function. The DL techniques can address the limitation of traditional ML algorithms since the traditional methods often require complex feature engineering, while the DL techniques have the capability to extract the features from input data automatically without human intervention. The performance of DL is significantly high at dealing with high non-linearity degrees of data points. Therefore, it is expected to improve cybersecurity trends, such as IDSs. This section represents the DL approach to tackle the problem of DDoS attacks in SDN networks.

1) The proposed DL Model: We use our previous model, i.e., DDoSnet [16], which composites from autoencoder and Recurrent Neural Network (RNN). The proposed model effectively detected DDoS attacks with great performance and low false alarms in comparison with traditional ML algorithms. However, in this article, we enhance the model performance by using LSTM, which is a specific type of RNN, instead of simple RNN in order to avoid the problem of vanishing gradient. Gradients are used to update the weight values of a neural network, as shown in equation (2). However, when a gradient value becomes extremely small, it does not contribute too much learning as it back propagates through time. The RNN suffers from small gradient updates, especially in the earlier layers. Thus, it is unable to keep the information for the long sequences.

$$New\ weight = weight - learning\ rate * gradient. \quad (2)$$

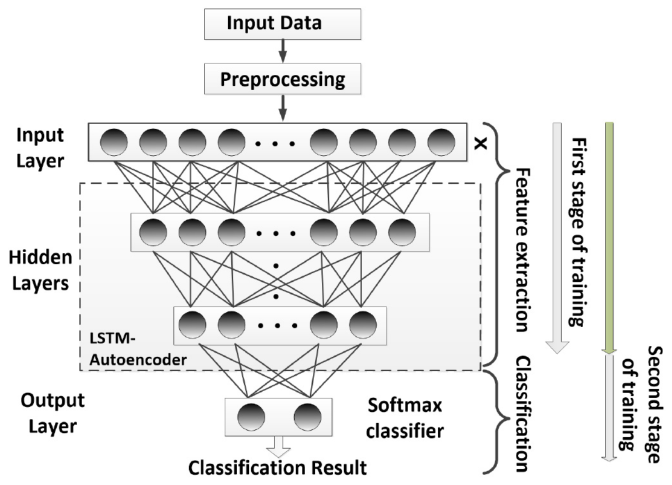


Fig. 10. The LSTM-autoecoder for DDoS attacks detection [16].

The overall structure of the DL approach is illustrated in Fig. 10. The model has two phases: (1) Pertaining phase, which uses unsupervised learning; (2) fine-tuning phase and uses supervised learning. The unsupervised learning is employed in the first stage without any labels to extract the discriminatory features of the raw data. The autoencoder takes the represented information in the original space and transforms it into another space. Each dense layer in the original autoencoder is replaced with an LSTM layer to improve the model performance. The nature of the network traffic is the key idea behind the using of LSTM in DL approach since the temporal correlation of the input data generates sequential traffic. Thus,

TABLE VI
LSTM-AUTOENCODER SPECIFICATIONS

Parameters	**Optimal values**
Hidden-layers	3
Number of channels (neurons)	32, 16, 8
Activation function	ReLU
Number of epochs	100
Loss function	MSE
Learning rate	10e-4
Optimizer type	Adam
Batch size	128

building the DL model with such techniques will eliminate the loss, as the output of any layer does not only depend on the current input but also based on the previous output.

2) Experimental Setting: Tuning hyper-parameters values is one of the significant challenges in DL training due to the lack of theoretical foundation. Unfortunately, there is no secret rule for choosing the optimal values of hyper-parameter. So, several experiments and combinations based on trial and error have been conducted to find the best number of network layers, number of neurons in each layer, number of iteration, batch size, etc. To demonstrate the best values of hyper-parameters, we analyse the performance of the DL model by testing different values of learning rate (λ), i.e., 0.0001, 0.001, 0.01, and 0.1. For each value of λ, we examine the model performance on other different parameters. For example, we test the impact of hidden layer numbers, size of channels in each layer, iteration, and the activation function on the entire performance of the DL approach. The best classifier accuracy is obtained when the number of hidden layers is equal to three. Repeating experiments several times, the results have shown that for the given data, the highest performance was obtained when we used the hyper-parameters as shown in Table VI. The value of hyper-parameters can be changed from one dataset to another. For simplicity, we used the same hyper-parameters as described in Table VI for all datasets since the variation in the results can be ignored.

At the encoder phase, the input dimensions are reduced to 32, 16, and 8 through the three hidden layers, respectively. The final output of the encoding phase is compressed input data. The decoded step is a reverse order of the encoded phase with the following number of channels: 8, 16, and 32, respectively. After building the model and finding the best values of weight and bias, the hierarchical features are obtained from unlabeled data. In the second stage, fine-tuning is used to optimise the network and train the highest layers of the network using labeled data (i.e., supervised learning). Finally, the output of the model is obtained by adding the softmax function at the output layer. The softmax layer generates an output in the range of (0, 1) for each output class, where all classes probability is equal to 1. In this work, we only examined samples from normal and DDoS attacks. So, binary classification is used to assign a digit 0 for normal and 1 for malicious or DDoS attacks. The `categorical cross-entropy` is used with the `softmax` layer, while the `Adam` optimiser is employed to update network weights iterative. We trained the model using 100 `epochs` and 128 for the `batch` size.

E. Feature Selection Algorithms

This section utilises the feature selection techniques to find the relevant features of DDoS attacks in each dataset separately. The feature selection methods explore the most relevant features for each class label while ignoring the redundant or irrelevant features. Therefore, training the detection model using few features can help to build a lightweight classifier, less prone to overfitting. Further, the lightweight model can be deployed easily in the network platform without causing any significant computational cost on the system resources [8]. While the strategy of finding the importance features is different from one feature selection algorithm to another, we used two different methods, i.e., Random Forest (RF) and Information Gain (IG).

1) Information Gain (IG): Is one of the most popular algorithms to compute how much each variable is contributing to the decision. It comes under the category of filter methods and identifies the importance of features based on the concept of information theory. A common measure for the information is Shannon entropy. The entropy quantifies the uncertainty of each feature according to its relevance in determining different classes. We can calculate the entropy for specific attack class *H(C)* using the following equation:

$$H(C) = -\sum_{i=0}^{n} \rho(i) log \rho(i). \tag{3}$$

The IG used a simple attribute rank by measuring information weight of each feature and eliminating the irrelevant features for each class label. A feature that has a small information gain, has also a low affect on the data classification and can be ignored without any degradation on the model performance. The IG for each individual input feature *F* in the dataset is obtained by calculating the reduction in the entropy according to the following equation:

$$IG(C; F) = H(C) - H(C|F) \tag{4}$$

where *IG(C;F)* is information gain of the feature *F*, taking into account the class features *C*, and *H(C|F)* is the average conditional entropy of *C*.

2) Random Forest (RF) [54]: Is widely used to solve the problem of individual Decision Trees (DTs) with a good predictive performance and less prone to overfitting. It is categorised under Embedded methods, which combines the filter and wrapper techniques. The key idea behind the RF is to measure how much each feature contributes on the prediction. If the change is large, this is an important variable. Similarly, when the change is small, this means the feature does not provide a significant information. The RF is a combination of several hundreds of decision trees (Fig. 11) which are built based on random observation from the dataset and random extraction of the features. The number of features can vary from one tree to another, and this immunes the model from overfitting since the trees are de-correlated. The RF calculates

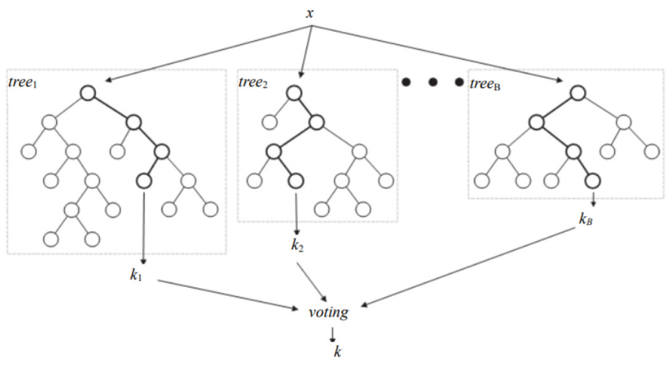


Fig. 11. The general architecture of random forest.

TABLE VII
THE BEST SUB SELECTED FEATURES BY RF AND IG

	Dataset	New Subset Features
RF	InSDN	1, 2, 5, 15, 18, 26, 31, 32, 33, 34
	CICIDS2017	3, 5, 7, 8, 10, 13, 22, 23, 25, 31
	CICIDS2018	3, 5, 7, 8, 10, 21, 23, 26, 31, 33
IG	InSDN	1, 2, 15, 18, 20. 27, 29, 30, 32,34
	CICIDS2017	3, 4, 8, 10, 12, 14, 22, 31, 32, 40
	CICIDS2018	5, 6, 7, 8, 10, 11, 12, 21, 31, 32

the importance of each variable using two ways. The first measure is based on the decrease of Gini impurity when a variable is chosen to split a node. The second measure is based on how much the accuracy decreases when the variable is excluded.

F. Feature Selection Process

Although the datasets hold the same number of features, the importance of these features is different from one dataset to another. Table VII shows that there are 7 common features (i.e., 3, 5, 7, 8, 10, 23, 31) between the CICIDS2017 and CICIDS2018 in case of using the RF algorithm. However, the InSDN dataset has only 1 common feature (i.e., feature number *5*) with the CICIDS2017, while it is combined with CICIDS2018 in 4 features (i.e., 5, 26, 31, 33). Similarly, in the case of using the IG algorithm, the CICIDS2017 and CICIDS2018 have five common features (i.e., 8, 10, 12, 31, 32), while the InSDN dataset has only one common feature (i.e., 32) with other two datasets. Additionally, the RF and IG are common for CICIDS2017 on five features (i.e., 3, 8, 10, 22, 31), while they are combined on six features (i.e., 5, 7, 8, 10, 21, 31) in case of CICIDS2018. In a similar way, six features (i.e., 1, 2, 15, 18, 32, 34) are common between RF and IG for the InSDN dataset.

The common features between the RF and IG indicate that the CICIDS2017 and CICIDS2018 have participated on three features (i.e., 8, 10, 31), while they have not shared any feature with the InSDN dataset. This is due to the fact that both CICIDS2017 and CICIDS2018 datasets were generated from the same infrastructure behaviour, i.e., the two environments are based on conventional networks. Thus, some features, which are more related to DDoS attacks are still common between the two datasets since the behaviour of the attack is similar in the two network environments. On the contrary, the SDN platform has different characteristics and operations functionality than the traditional network. In addition, separating the control plane from the data plane produces new DDoS attacks, which have different behaviour from those reported in other networks.

On the other side, not only the importance of the features varies from one network to another, but the identity of these features can also vary from one environment to another. For example, the flow duration in conventional networks indicates the length of connections in seconds between the source and destination hosts, while the flow duration in SDN networks indicates the time during which the flow entry remains in the switch flow table [45]. Therefore, we can see the duration feature is more specific for DDoS attacks in SDNs. During the DDoS attacks, the malicious flow spends a long duration in the switch flow table compared to the legitimate traffic [45]. Thus, the flow, which remains active for a larger duration, is a good indicator for malicious DDoS attacks. Moreover, the attacker can flood the SDN network with a high amount of useless flows using spoofed IP addresses. Thus, the average number of packets per flow will decrease during the DDoS attacks since the attacker aims to flood the flow switches, consuming its flow tables space without sending data packets. So, some attributes like *'Flow_IAT_Max'* are important features to identify the malicious DDoS traffic. Such attributes will decrease in case of attacks, while it has a high value for normal traffic.

V. Experimental Results and Findings

A. The Evaluation Metrics

The performance of the model is evaluated using the most popular performance measures like the accuracy, precision, recall, and F-score metrics and are computed as the following equations.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{6}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{7}$$

$$\text{F-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{8}$$

where, True Positive (TP) and True Negative (TN) represent the correctly predicted values, while False Positive (FP) and False Negative (FN) indicate misclassified events.

B. Analysis Tools

We evaluated the performance of the DDoS attack detection model with Python programming language using Keras Library with Tensorflow backend. The testbed hardware and software parameters are depicted in Table VIII.

C. Experimental Results

This section discusses the experimental results of the proposed approach. Table IX shows the percentage of Precision, Recall and f1-score for datasets with a different subset of features. The model provided the highest results when

TABLE VIII
EXPERIMENTAL ENVIRONMENT

Operating System	Windows 10 pro 64-bit
Memory	16 GB
CPU	Intel(R) UHD Graphics 620, I7-8650U CPU @ 1.90GHz (8 cores), 2.1GHz
Python	3.7.0
Keras	2.4.2
Tensorflow	2.2.0

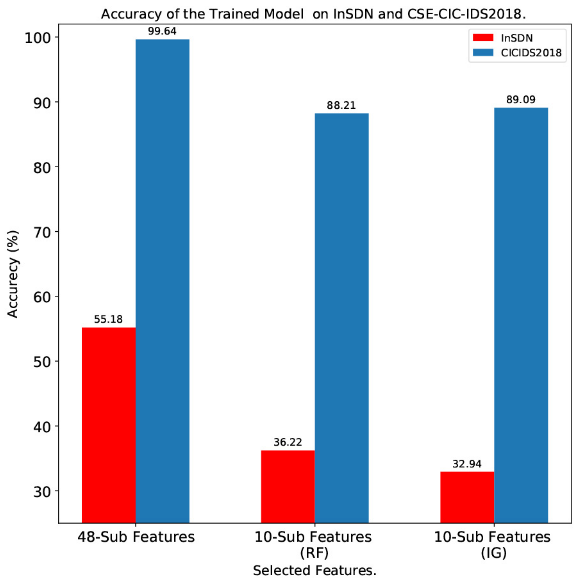


Fig. 12. Accuracy of the Trained Model by CICIDS2017 on InSDN and CICIC2018.

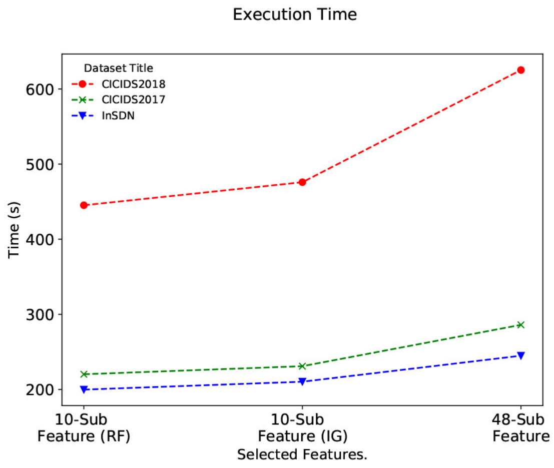


Fig. 13. Execution time.

48 sub-features are used in the training process, while the performance of the model is slightly declined when only 10 sub-features are used for both RF and IG algorithms. However, the decline in the performance can be ignored, so we can build a lightweight model with less number of features. Building a lightweight model will consume less amount of resources and make it more suitable for the SDN platform. It can also be noticed that the overall performance of the IG algorithm is slightly higher than the RF method.

We also validate our reclaim and demonstrate how the model performance can be significantly decreased when we evaluate it on different datasets produced from different environments.

In the training phase, we train the proposed DL approach using CICIDS2017 dataset, but we analyse its performance on the test portion of InSDN and CICIDS2018 datasets. We firstly train and test the model using all 48 sub features, and later we only use the best 10 features from RF and IF, which were selected earlier from the perspective of the CICIDS2017 dataset. The obtained results are described on Table X and Fig. 12. The results show that when we test the model performance on CICIDS2018 dataset using sub of 48 features, the evaluation metrics are significantly high and greatly near to those reported in Table IX. However, the performance is sharply declined on InSDN dataset. The reported accuracy for both CICIDS2018 and InSDN datasets are 99.61% and 55.18%, respectively. Moreover, when the model is trained on only 10 features, the performance for both datasets is largely reduced, but the decline is significantly huge for InSDN compared to CICIDS2018. The model failed to identify any DDoS entity for InSDN dataset. The overall accuracy is 88.21% and 36.22% for CICIDS2018 and InSDN dataset in the case of the RF method, while the overall accuracy is 89.09% and 32.94% when the IG method is used.

To further evaluate the performance of the model and to indicate the effect of the feature selection process, we represent the execution time of the model for all datasets, as shown in Fig. 13. The graph shows that the execution time of the model is relativity high for CICIDS2018, while it is low for InSDN. This is due to the fact that the size of samples in CICIDS2018 is very large compared to other datasets. It is also noticed that the model spent a long time for training when all 48 sub-features are used, while the execution time is relativity small in the case of RF but slightly higher for IG.

D. Comparative Analysis With State-of-the-Art

1) Comparative Analysis With DDoSnet: We further evaluate the model with our previous DDoSnet [16] approach. In [16], the DDoSnet was compared with several ML algorithms, such as Naive-Bayes (NB),Logistic Regression (LR), DT, RF, and SVM. The obtained results for different methods are depicted in Table XI. The results show the potential of the DL techniques for DDoS attack detection compared with the traditional algorithms. However, using the LSTM in the current model provides a high accuracy compared to the simple RNN algorithm.

We additionally estimate the performance of our model by calculating the lower and upper bounds of the confidence interval. The confidence interval is a way of quantifying the uncertainty of an estimate [55]. The lower and upper bounds within which the statistic can vary are usually referred to as the *margin of error*. It provides a very clear understanding of how the true result may differ from the estimated results and how much more or less than the stated percentage the reality might be. The experimental results showed that the lower and upper bounds on the model's classification accuracy are 0.001, 0.002, respectively. While, the lower and upper bounds on the previous DDoSnet classification accuracy are 0.002,

TABLE IX
EVALUATION METRICS OF 48 AND 10 SUB-SET FEATURES

	Dataset	Precision (%)		Recall (%)		f1-score (%)	
		Normal	DDoS	Normal	DDoS	Normal	DDoS
48-Sub Feature	InSDN	99.93	99.96	99.93	99.96	99.93	99.96
	CICIDS2017	99.79	99.97	99.96	99.84	99.88	99.90
	CICIDS2018	100	99.99	99.99	100	99.99	99.99
	InSDN	99.90	99.89	99.82	99.94	99.86	99.92
10-Sub Feature (RF)	CICIDS2017	98.20	99.20	98.96	98.62	98.57	98.91
	CICIDS2018	99.99	99.98	99.98	99.99	99.98	99.98
	InSDN	99.96	99.91	99.95	99.93	99.95	99.92
10-Sub Feature (IG)	CICIDS2017	99.84	99.30	99.07	99.88	99.46	99.59
	CICIDS2018	100	99.99	99.99	100	99.99	99.99

TABLE X
RESULTS OF THE TRAINED MODEL BY CICIDS2017 ON INSDN AND CICIDS2018

	Dataset	Precision (%)		Recall (%)		f1-score (%)	
		Normal	DDoS	Normal	DDoS	Normal	DDoS
48-Sub Feature	InSDN	44.14	85.59	89.41	35.72	59.11	50.41
	CICIDS2018	99.73	99.53	99.55	99.72	99.64	99.63
10-Sub Feature (RF)	InSDN	36.22	20.41	99.98	0.10	53.18	0.21
	CICIDS2018	81.42	99.57	99.69	76.18	89.64	86.32
10-Sub Feature (IG)	InSDN	34.06	0	90.93	0	49.56	0
	CICIDS2018	81.37	99.31	99.49	76.14	89.52	86.19

TABLE XI
COMPARISON TO DDOSNET MODEL [16]

Techniques	Accuracy (%)
NB	57
DT	77
RF	86
SVM	93
LR	95
DDoSNet	99
Proposed (InSDN)	99.95

0.003, respectively. Therefore, the classification error of the represented classifier is less than the error described in the DDoSnet, which indicates the high efficiency of the proposed model.

2) Comparative Analysis on CICIDS2017 Dataset: We further carried out a comparative analysis with two different studies [44] and [43] on CICICI2017. Table XII represents a comparative analysis of the two studies with the proposed model. In [44] and [43], several ML algorithms have been employed with the feature selection methods. It is noticed that our DL approach provided the highest accuracy compared to other work. The reported accuracy of the proposed model with IG and RF selection methods is 99.50% and 98.76%, respectively.

VI. NETWORK PERFORMANCE ANALYSIS

This section provides a detailed analysis of the proposed DL approach on the performance of the SDN controller. We used the same framework of [46] to evaluate the impact of the proposed model on the network performance in terms of throughput and latency. The *Cbench* tool[2] is utilised to evaluate the performance of the controller with various numbers of OpenFlow switches. The *Cbench* tool is used to evaluate the overheads of the DL model on the SDN controller. It provides two different options to test the throughput and latency as the follow:

1) In the throughput mode, *Cbench* generates a stream of `packet-In` message to the SDN controller and then records the `packet-Out` message that have been received in a period of time. Calculating the sending and receiving stream provides a good indication of the average number of flows that the controller can handle for each switch per second.
2) In the latency mode, the *Cbench* sends a `packet-In` message to the controller and waits for the response before sending the next packet. Hence, we can find average number of milliseconds that a flow consumes to be installed in each switch.

The model is written in Python programming language and embedded on top of the SDN controller as an application layer. We compare our model performance in terms of throughput and latency on the Ryu controller after training it on three various datasets. The experiments are conducted on a Linux virtual machine running 64-bit Ubuntu 18.04 LTS, 8 GB of RAM, Core-i7 CPU, and installed on a VMware workstation 15 Pro.

A. Throughput Results

The throughput represents the size of the packets that the controller can handle per second. The throughput size is varied

[2]https://github.com/trema/cbench

TABLE XII
COMPARATIVE ANALYSIS ON CICIDS2017 DATASET

Solution	Feature Selection Method	Accurecy (%)
Bindra et. al. [44] (CICIDS2017)	Select Percentile (K=15),	96.1
	RFE (LinearSVC and Components=15)	82.4
	RFE with Lasso (15 features)	96.65
	PCA (n=25)	95.6
	SelectFromModel (12 features)	96.5
Abdulrahman et. al. [43] (CICIDS2017)	IG (10)	86.80
Proposed (CICIDS2017)	**RF (10)**	**98.76**
	IG (10)	**99.50**

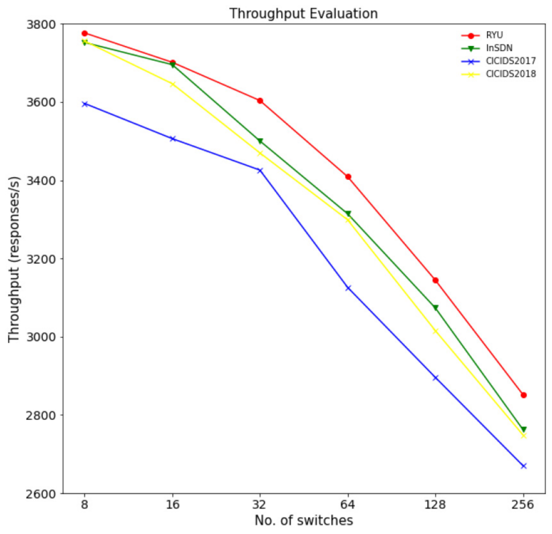


Fig. 14. Throughput Evaluation.

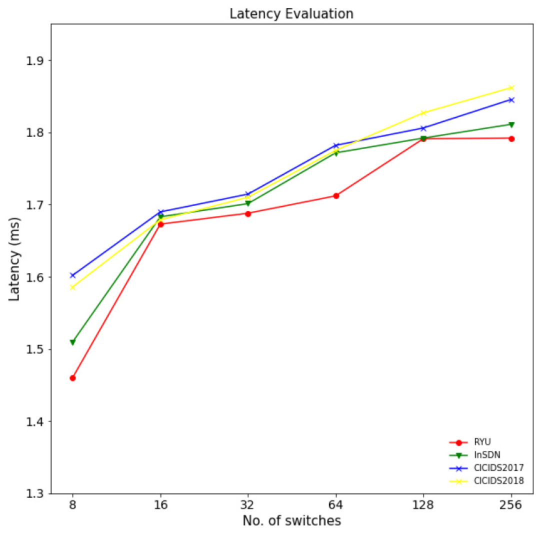


Fig. 15. Latency Evaluation.

from one SDN controller to another. For simplicity, we test the effect of the detection model on the Ryu controller. Figure 14 illustrates the throughput of the controller with our model using different datasets. The graph shows that the throughput of the running standalone Ryu controller is limited at 3800 packet/s, which is very low compared to other controllers [56], [57]. However, the standalone Ryu Controller provides a high throughput compared to the embedded DL model. Therefore, we take it as a baseline for evaluating the detection model. The throughput of the Ryu controller and the embedded model are declined with increasing the number of switches. However, the performance of the model is varied according to the used dataset. The decline in the throughput can be ignored in small network typologies when the model is trained on InSDN or CICIDS2018, while the drop is significantly high for the CICIDS2017.

The throughput of the model in the case of using the InSDN dataset is dropped by about 2.86% and 3.1% when the number of switches increases from 32 to 256, respectively. Compared with the CICIDS2018, the throughput decreases by 3.7 and 4.1% and is significantly reduced by 2.8% and 6.37% for CICIDS2017. It can noticed that the training dataset not only plays a vital role in the potential of the classifier capability; but it is also effective in determining the performance of the model inside the network. We can see that the model with InSDN data provides less overhead on the controller compared to other datasets.

B. Latency Evaluation

The latency test is represented in Fig. 15. Similar to the aforementioned throughput results, the latency increases with the increase of the topology size. The standalone controller has less latency compared to the embedded model, regardless the dataset used. Integrating the security model with the controller can quite increase the controller latency. However, the model with InSDN dataset experienced a small latency, followed by CICIDS2018, while the trained model on CICIDS2017 has the highest latency, almost for all networks typologies.

The above results indicate that there is a trade-off between network performance and security. Implementing security can comprehensively decrease network performance. Therefore, tuning the network relies on the IT operations to find the best adjustments based on their requirements, either by enhancing the network security with a little delay or keep it fast [58].

VII. DISCUSSION AND LIMITATION

Although the SDN is a promising solution for anomaly detection systems, the SDN itself can be a target for several attack threats. Unfortunately, all SDN layers are susceptible to DDoS attacks, which can easily consume its resources and prevent or even delay the network services for legitimate users. Therefore, eliminating the impact of these attacks has gained significant attention from the research community in the last decade. Instantaneously, there is an increasing direction of using machine and deep learning techniques for anomaly detection systems to solve the problem of DDoS attacks in SDNs. However, the quality of the training dataset is a key pillar of any model efficiency.

On the other hand, one of the main challenges, which seriously hinder the performance of the ML/DL models is the problem of overfitting. The model can effectively perform very well during the training but fails to display a good tendency with the unseen data. There are many reasons that can cause this problem such as, the complexity of the model and the low amount of data used to create a suitable approach. Thus, the best practice to test the efficacy of intrusion detection models is to evaluate how it can work with new data that have never been seen before during the training. This is what we investigated and successfully achieved in this work.

Nonetheless, the majority of the current anomaly detection techniques in SDNs have been evaluated using a dataset generated based on IP-traditional networks and not from SDNs. However, the SDN platform generates new attack vectors that did not exist before in traditional networks. Thus, training the detection model using an improper dataset can deceive the classifier model and make it easily prone to overfitting. In addition, the behaviour of the attacks is different from one environment to another. For example, the attacker can exploit the operation of the SDN and employ some existing attacks such as "Port scan" and "IP sweep" to overwhelm the controller with a heavy volume of unknown traffic, creating a new DDoS attack vector [59]. However, conventional detection systems can easily identify "Port scan" and "IPsweep", but the functionality of these attacks are different in SDNs, i.e., work as DDoS. Moreover, DDoS attacks are rapidly evolving threat and can cause a crucial impact on the performance of network services running over SDN [60]. Hence, the availability and response time of SDN services are significantly degraded at presence of attacks. In this article, we demonstrate how the importance of features is being changed from one dataset to another, regardless of the fact that the used datasets have the same attack classes or a similar number of features. However, some of attributes that are widely used for model classifiers on IP based networks can have less impact in the SDN and vice versa [61]. Based on the analysis and the above results, we showed that the behaviour and operation of the SDNs are varied from other networks. Thus, the structure of the new platform should be taken due to the design of anomaly detection systems.

Likewise, our proposed model is experienced to some limitations which are listed below:

- We test the performance of the model using only one SDN controller; however, the throughput and latency are varied from one controller to another. Thus, several controllers should be examined for fair awareness and to represent how the embedded model can work efficiently with other controllers.
- We trained and evaluated the DL model in offline mode using virtual simulation without implementing a physical SDN networks. However, the detection of attacks online is very important to understand how this IDS can handle the intrusion in real-time.
- In this article, we employed the DL model to produce a lightweight model against DDoS attacks. Despite the DDoS attacks being one of the most dangerous attacks in the SDN, the SDN is vulnerable to many other attacks that can compromise its normal operation. In the near future, we will train the DL model to consider new attacks in the SDN. In addition, a new experimental test should be used to classify the data categories into normal or attack classes, i.e., using multi-classification instead of binary classification.
- The adversaries can actively adapt and modify their threat models to learn the decision boundary of the anomaly detector. They aim to compromise the integrity of anomaly detectors by reducing the confidence and modifying the input (an anomalous sample) in order to output (nominal class) by the detector [62]. Therefore, understanding the adversary threat model will help avoid mistakes and reduce the false positive alarms of the anomaly detectors. However, the attack methodology, which adversarial examples reside is beyond the scope of this paper. The interested reader can refer to [62]–[65] for more information regarding the general strategies that an attacker can use against any anomaly detector.

VIII. Conclusion

Training the network intrusion detection system using a high-dimensional dataset increases the complexity of proposed classifier, which result in excessive training and classification time. The pre-processing feature selection methods play an essential role in identifying the important features from the original dataset, and this would help to improve the classification accuracy and avoid the curse of high computational complexity. The aim of this work is to reduce the redundant or irrelevant features without any significant impact on the classification accuracy. We have selected 10 features out of available 48 features using two common feature selection methods IG and RF. A modified DL model based on LSTM-Autoencoder was used for experimental purposes, while the DDoS attacks were considered as a case study. Our approach provides a high detection rate and presents a more efficient better time to build the model. We further tested the trained model on the performance of the SDN controller to evaluate how the used dataset can impact on the performance of the SDN controller. The results showed that the proposed approach does not deteriorate the network performance. In our future work, we will analyse new attack classes for the test evaluation. Also, we plan to apply our proposed model on real SDN network in order to understand how this IDS can handle the intrusion in real-time.

References

[1] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[2] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 27–51, 1st Quart., 2015.

[3] S. Shin and G. Gu, "Attacking software-defined networks: A first feasibility study," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, 2013, pp. 165–166.

[4] Ö. Kasim, "An efficient and robust deep learning based network anomaly detection against distributed denial of service attacks," *Comput. Netw.*, vol. 180, Oct. 2020, Art. no. 107390.

[5] N. Garcia, T. Alcaniz, A. González-Vidal, J. B. Bernabe, D. Rivera, and A. Skarmeta, "Distributed real-time SlowDoS attacks detection over encrypted traffic using artificial intelligence," *J. Netw. Comput. Appl.*, vol. 173, Jan. 2021, Art. no. 102871.

[6] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *J. Netw. Comput. Appl.*, vol. 84, pp. 25–37, Apr. 2017.

[7] K. Bouzoubaa, B. Nsiri, and Y. Taher, "Predicting DOS-DDOS attacks: Review and evaluation study of feature selection methods based on wrapper process," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 5, pp. 131–145, 2021.

[8] D. Kurniabudi, D. Stiawan, M. Y. B. Idris, A. M. Bamhdi, and R. Budiarto, "CICIDS-2017 dataset feature analysis with information gain for anomaly detection," *IEEE Access*, vol. 8, pp. 132911–132921, 2020.

[9] A. Bommert, X. Sun, B. Bischl, J. Rahnenführer, and M. Lang, "Benchmark for filter methods for feature selection in high-dimensional classification data," *Comput. Stat. Data Anal.*, vol. 143, Mar. 2020, Art. no. 106839.

[10] J. Tang, S. Alelyani, and H. Liu, "Feature selection for classification: A review," in *Data Classification: Algorithms Applications*. Boca Raton, FL, USA: CRC Press, 2014, p. 37.

[11] H. Polat, O. Polat, and A. Cetin, "Detecting DDoS attacks in software-defined networks through feature selection methods and machine learning models," *Sustainability*, vol. 12, no. 3, p. 1035, 2020.

[12] M. Suresh and R. Anitha, "Evaluating machine learning algorithms for detecting DDoS attacks," in *Proc. Int. Conf. Netw. Security Appl.*, 2011, pp. 441–452.

[13] E. Balkanli, A. N. Zincir-Heywood, and M. I. Heywood, "Feature selection for robust backscatter DDoS detection," in *Proc. IEEE 40th Local Comput. Netw. Conf. Workshops (LCN Workshops)*, 2015, pp. 611–618.

[14] M. S. El Sayed, N.-A. Le-Khac, and A. D. Jurcut, "InSDN: A novel SDN intrusion dataset," *IEEE Access*, vol. 8, pp. 165263–165284, 2020.

[15] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. ICISSP*, 2018, pp. 108–116.

[16] M. S. El Sayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "DDoSNet: A deep-learning model for detecting network attacks," in *Proc. IEEE 21st Int. Symp. World Wireless Mobile Multimedia Netw. (WoWMoM)*, 2020, pp. 391–396.

[17] J. C. C. Chica, J. C. Imbachi, and J. F. B. Vega, "Security in SDN: A comprehensive survey," *J. Netw. Comput. Appl.*, vol. 159, Jun. 2020, Art. no. 102595.

[18] H. Griffioen, K. Oosthoek, P. van der Knaap, and C. Doerr, "Scan, test, execute: Adversarial tactics in amplification DDoS attacks," in *Proc. 2021 ACM SIGSAC Conf. Comput. Commun. Security*, 2021, pp. 940–954.

[19] M. S. El Sayed, N.-A. Le-Khac, M. A. Albahar, and A. Jurcut, "A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique," *J. Netw. Comput. Appl.*, vol. 191, Oct. 2021, Art. no. 103160.

[20] N. Z. Bawany, J. A. Shamsi, and K. Salah, "DDoS attack detection and mitigation using SDN: Methods, practices, and solutions," *Arab. J. Sci. Eng.*, vol. 42, no. 2, pp. 425–441, 2017.

[21] T. Ubale and A. K. Jain, "Taxonomy of DDoS attacks in software-defined networking environment," in *Proc. Int. Conf. Futuristic Trends Netw. Commun. Technol.*, 2018, pp. 278–291.

[22] Y. Qian, W. You, and K. Qian, "Openflow flow table overflow attacks and countermeasures," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, 2016, pp. 205–209.

[23] J. Singh and S. Behal, "Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions," *Comput. Sci. Rev.*, vol. 37, Aug. 2020, Art. no. 100279.

[24] R. U. Rasool, U. Ashraf, K. Ahmed, H. Wang, W. Rafique, and Z. Anwar, "Cyberpulse: A machine learning based link flooding attack mitigation system for software defined networks," *IEEE Access*, vol. 7, pp. 34885–34899, 2019.

[25] Z. Li, W. Xing, S. Khamaiseh, and D. Xu, "Detecting saturation attacks based on self-similarity of OpenFlow traffic," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 1, pp. 607–621, Mar. 2020.

[26] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," *Comput. Netw.*, vol. 62, pp. 122–136, Apr. 2014.

[27] L. Tan, Y. Pan, J. Wu, J. Zhou, H. Jiang, and Y. Deng, "A new framework for DDoS attack detection and defense in SDN environment," *IEEE Access*, vol. 8, pp. 161908–161919, 2020.

[28] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[29] A. Alshamrani, A. Chowdhary, S. Pisharody, D. Lu, and D. Huang, "A defense system for defeating DDoS attacks in SDN based networks," in *Proc. 15th ACM Int. Symp. Mobility Manage. Wireless Access*, 2017, pp. 83–92.

[30] M. S. El Sayed, N.-A. Le-Khac, and A. D. Jurcut, "Dealing with COVID-19 network traffic spikes [cybercrime and forensics]," *IEEE Security Privacy*, vol. 19, no. 1, pp. 90–94, Jan./Feb. 2021.

[31] P. Kumar, M. Tripathi, A. Nehra, M. Conti, and C. Lal, "SAFETY: Early detection and mitigation of TCP SYN flood utilizing entropy in SDN," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 4, pp. 1545–1559, Dec. 2018.

[32] K. Kalkan, L. Altay, G. Gür, and F. Alagöz, "JESS: Joint entropy-based DDoS defense scheme in SDN," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2358–2372, Oct. 2018.

[33] S. Yu, J. Zhang, J. Liu, X. Zhang, Y. Li, and T. Xu, "A cooperative DDoS attack detection scheme based on entropy and ensemble learning in SDN," *EURASIP J. Wireless Commun. Netw.*, vol. 90, no. 1, pp. 1–21, 2021.

[34] A. Mishra, N. Gupta, and B. Gupta, "Defense mechanisms against DDoS attack based on entropy in SDN-cloud using POX controller," *Telecommun. Syst.*, vol. 77, no. 1, pp. 47–62, 2021.

[35] K. S. Sahoo, D. Puthal, M. Tiwary, J. J. Rodrigues, B. Sahoo, and R. Dash, "An early detection of low rate DDoS attack to SDN based data center networks using information distance metrics," *Future Gener. Comput. Syst.*, vol. 89, pp. 685–697, Dec. 2018.

[36] R. Wang, Z. Jia, and L. Ju, "An entropy-based distributed DDoS detection mechanism in software-defined networking," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, vol. 1, 2015, pp. 310–317.

[37] M. S. El Sayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Detecting abnormal traffic in large-scale networks," in *Proc. Int. Symp. Netw. Comput. Commun. (ISNCC)*, 2020, pp. 1–7.

[38] M. Said El Sayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Network anomaly detection using LSTM based autoencoder," in *Proc. 16th ACM Symp. QoS Security Wireless Mobile Netw.*, 2020, pp. 37–45.

[39] S. Dong and M. Sarem, "DDoS attack detection method based on improved KNN with the degree of DDoS attack in software-defined networks," *IEEE Access*, vol. 8, pp. 5039–5048, 2019.

[40] L. Yang and H. Zhao, "DDoS attack identification and defense using SDN based on machine learning method," in *Proc. 15th Int. Symp. Pervasive Syst., Algorithms Netw. (I-SPAN)*, 2018, pp. 174–178.

[41] Y. Yu, L. Guo, Y. Liu, J. Zheng, and Y. Zong, "An efficient SDN-based DDoS attack detection and rapid response platform in vehicular networks," *IEEE Access*, vol. 6, pp. 44570–44579, 2018.

[42] J. A. Pérez-Díaz, I. A. Valdovinos, K.-K. R. Choo, and D. Zhu, "A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning," *IEEE Access*, vol. 8, pp. 155859–155872, 2020.

[43] A. A. Abdulrahman and M. K. Ibrahem, "Evaluation of DDoS attacks detection in a CICIDS2017 dataset based on classification algorithms," *IRAQI J. Inf. Commun. Technol.*, vol. 1, no. 3, pp. 49–55, 2018.

[44] N. Bindra and M. Sood, "Evaluating the impact of feature selection methods on the performance of the machine learning models in detecting DDoS attacks," *Sci. Technol.*, vol. 23, no. 3, pp. 250–261, 2020.

[45] N. Ahuja, G. Singal, D. Mukhopadhyay, and N. Kumar, "Automated DDOS attack detection in software defined networking," *J. Netw. Comput. Appl.*, vol. 187, Aug. 2021, Art. no. 103108.

[46] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep recurrent neural network for intrusion detection in SDN-based networks," in *Proc. 4th IEEE Conf. Netw. Softwarization Workshops (NetSoft)*, 2018, pp. 202–206.

[47] C. Li *et al.*, "Detection and defense of DDoS attack–based on deep learning in OpenFlow-based SDN," *Int. J. Commun. Syst.*, vol. 31, no. 5, 2018, Art. no. e3497.

[48] S. Haider *et al.*, "A deep CNN ensemble framework for efficient DDoS attack detection in software defined networks," *IEEE Access*, vol. 8, pp. 53972–53983, 2020.

[49] M. P. Novaes, L. F. Carvalho, J. Lloret, and M. L. Proença, Jr., "Adversarial deep learning approach detection and defense against DDoS attacks in SDN environments," *Future Gener. Comput. Syst.*, vol. 125, pp. 156–167, Dec. 2021.

[50] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Comput. Security*, vol. 86, pp. 147–167, Sep. 2019.

[51] M. T. A. Abdullah, J. Lloret, A. Cánovas Solbes, and L. García-García, "Survey of transportation of adaptive multimedia streaming service in Internet," *Netw. Protocols Algorithms*, vol. 9, nos. 1–2, pp. 85–125, 2017.

[52] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related," in *Proc. 2nd Int. Conf. Inf. Syst. Security Privacy (ICISSP)*, 2016, pp. 407–414.

[53] P. Krishnan, S. Duttagupta, and K. Achuthan, "VARMAN: Multi-plane security framework for software defined networks," *Comput. Commun.*, vol. 148, pp. 215–239, Dec. 2019.

[54] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[55] J. H. Steiger and R. T. Fouladi, "Noncentrality interval estimation and the evaluation of statistical models," in *What If There Were no Significance Tests*. London, U.K.: Routledge, 2016, pp. 197–229.

[56] A. T. Tang, "Software defined networking: Network intrusion detection system," Ph.D. dissertation, Dept. Electron. Elect. Eng., Univ. Leeds, Leeds, U.K., 2019.

[57] M. M. Isa and L. Mhamdi, "Native SDN intrusion detection using machine learning," in *Proc. IEEE 8th Int. Conf. Commun. Netw. (ComNet)*, 2020, pp. 1–7.

[58] M. A. Albahar, "Recurrent neural network model based on a new regularization technique for real-time intrusion detection in SDN environments," *Security Commun. Netw.*, vol. 2019, Nov. 2019, Art. no. 8939041.

[59] M. Conti, A. Gangwal, and M. S. Gaur, "A comprehensive and effective mechanism for DDoS detection in SDN," in *Proc. IEEE 13th Int. Conf. Wireless Mobile Comput. Netw. Commun. (WiMob)*, 2017, pp. 1–8.

[60] Q. Niyaz, W. Sun, and M. Alam, "Impact on SDN powered network services under adversarial attacks," *Procedia Comput. Sci.*, vol. 62, pp. 228–235, Aug. 2015.

[61] R. Santos, D. Souza, W. Santo, A. Ribeiro, and E. Moreno, "Machine learning algorithms to detect DDoS attacks in SDN," *Concurrency Comput. Pract. Exp.*, vol. 32, no. 16, 2020, Art. no. e5402.

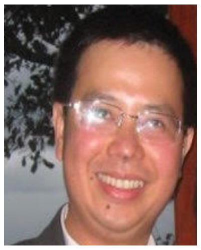
[62] A. Kuppa, S. Grzonkowski, M. R. Asghar, and N.-A. Le-Khac, "Black box attacks on deep anomaly detectors," in *Proc. 14th Int. Conf. Avail. Rel. Security*, 2019, pp. 1–10.

[63] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognit.*, vol. 84, pp. 317–331, Dec. 2018.

[64] Q. Liu, P. Li, W. Zhao, W. Cai, S. Yu, and V. C. Leung, "A survey on security threats and defensive techniques of machine learning: A data driven view," *IEEE Access*, vol. 6, pp. 12103–12117, 2018.

[65] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symp. Security Privacy (SP)*, 2016, pp. 582–597.

Mahmoud Said El Sayed received the B.E. degree in electronics and communication engineering from Zagazig University, Zagazig, Egypt, in 2007, the M.E. degree in information security from Nile University, Cairo, Egypt, in 2018, and the first Diploma degree in management of innovation from the DIT Institute, Dublin, Ireland, and the second Diploma degree in networking from the ITI Institute, Cairo. He is currently pursuing the Ph.D. degree with the School of Computer Science, UCD, Dublin. Besides, a set of international professional certificates in the computer networks, security, and IT fields from Cisco, IBM, Huawei, and VMware systems. He has worked for several years in the industry through Huawei and IBM Company in area of computer network and security. His research interests include, but not limited to computer networks, cybersecurity, cyber threat and attacks for android, artificial intelligence, routing protocols, security for the Internet of Things, and software-defined networks.



Nhien-An Le-Khac (Senior Member, IEEE) received the Ph.D. degree in computer science from the Institut National Polytechnique de Grenoble, France, in 2006. He is a Lecturer with the School of Computer Science, University College Dublin (UCD), Ireland. He is currently the Programme Director of UCD M.Sc. programme in Forensic Computing and Cybercrime Investigation, an international programme for the law enforcement officers specializing in cybercrime investigations. To date, more than 1000 students from 60 countries in five continents have graduated from this FCCI programme. He is also the Co-Founder of UCD-GNECB Postgraduate Certificate in fraud and e-crime investigation. He was a Research Fellow with Citibank, Ireland (Citi). His research interests span the area of cybersecurity and digital forensics, machine learning for security, fraud and criminal detection, cloud security and privacy, and high-performance computing. Since 2013, he has collaborated on many research projects as a principal/co-PI/funded investigator. He has published more than 150 scientific papers in peer-reviewed journal and conferences in related research fields. He is an active chair as well as a reviewer for many key conferences and journals in related disciplines.

Marianne A. Azer received the B.Sc., M.Sc., and Ph.D. degrees from the Faculty of Engineering, Electronics and Communications Department, Cairo University. She is an Associate Professor with the National Telecommunication Institute, Nile University, Cairo, Egypt. She is also the Director of the Information Center, National Telecommunication Institute. Her research interests include network security, security in wireless networks, Internet of Things privacy and security, and cloud security and privacy. She has been a Board Member of the Financial Regulatory Authority since May 2021. She is a former member of the Egyptian Parliament and a former advisor to the Ministry of Communication and Information Technology for strategic initiatives. She has been the Vice President of Information Systems Audit and Control Association (ISACA) Board in Egypt since 2019. Throughout her career, she held several positions, either academic or managerial in several universities and organizations. To mention a few, the Ministry of Communication and Information Technology, the National Telecommunication Institute, Nile University, Cairo University, The American University in Cairo, French University, the Arab Academy for Science and Technology, and Maritime Transport. She was the President of ISACA Board in Egypt 2018–2020, a member of the Global Advisory Board on Emerging Technologies (ISACA) 2020–2021, and also a member of the Global Advisory Board for Facebook Community Leadership Program 2018–2019. She received many awards and recognitions both on the international and national levels. She is a member of international and national organizations in diverse fields, such as telecommunications, politics, women, science, technology, culture, angel investment, and governance.

Anca D. Jurcut received the B.Sc. degree in computer science and mathematics from the West University of Timisoara, Romania, in 2007, and the Ph.D. degree in security engineering from the University of Limerick (UL), Ireland, in 2013 funded by the Irish Research Council for Science Engineering and Technology. She has been an Assistant Professor with the School of Computer Science, University College Dublin (UCD), Ireland, since 2015. She worked as a Postdoctoral Researcher with UL as a member of the Data Communication Security Laboratory and as a Software Engineer with IBM, Dublin, Ireland, in the area of data security and formal verification. Her research interests include security protocols design and analysis, automated techniques for formal verification, network security, attack detection and prevention techniques, security for the Internet of Things, and applications of blockchain for security and privacy. She has several key contributions in research focusing on detection and prevention techniques of attacks over networks, the design and analysis of security protocols, automated techniques for formal verification, and security for mobile-edge computing. More Info: https://people.ucd.ie/anca.jurcut.