Entropy based features distribution for anti-DDoS model in SDN

Ujjan, Raja Majid Ali; Pervez, Zeeshan; Dahal, Keshav; Khan, Wajahat Ali; Khattak, Asad Masood; Hayat, Bashir

Published in:
Sustainability (Switzerland)

DOI:
10.3390/su13031522

Published: 01/02/2021

Document Version
Publisher's PDF, also known as Version of record

sustainability

MDPI

Article

Entropy Based Features Distribution for Anti-DDoS Model in SDN

Raja Majid Ali Ujjan [1], Zeeshan Pervez [1,*], Keshav Dahal [1], Wajahat Ali Khan [2], Asad Masood Khattak [3] and Bashir Hayat [4]

[1] School of Computing, Engineering and Physical Sciences, University of the West of Scotland, Paisley PA1 2BE, UK; raja.vjjan@uws.ac.uk (R.M.A.U.); keshav.dahal@uws.ac.uk (K.D.)
[2] College of Engineering and Technology, University of Derby, Derby DE22 3AW, UK; w.khan@derby.ac.uk
[3] College of Technological Innovation, Zayed University, P.O. Box 144534, Abu Dhabi, United Arab Emirates; asad.khattak@zu.ac.ae
[4] Institute of Management Sciences, Peshawar 54600, Pakistan; bashir.hayat@imsciences.edu.pk
* Correspondence: zeeshan.pervez@uws.ac.uk

Abstract: In modern network infrastructure, Distributed Denial of Service (DDoS) attacks are considered as severe network security threats. For conventional network security tools it is extremely difficult to distinguish between the higher traffic volume of a DDoS attack and large number of legitimate users accessing a targeted network service or a resource. Although these attacks have been widely studied, there are few works which collect and analyse truly representative characteristics of DDoS traffic. The current research mostly focuses on DDoS detection and mitigation with predefined DDoS data-sets which are often hard to generalise for various network services and legitimate users' traffic patterns. In order to deal with considerably large DDoS traffic flow in a Software Defined Networking (SDN), in this work we proposed a fast and an effective entropy-based DDoS detection. We deployed generalised entropy calculation by combining Shannon and Renyi entropy to identify distributed features of DDoS traffic—it also helped SDN controller to effectively deal with heavy malicious traffic. To lower down the network traffic overhead, we collected data-plane traffic with signature-based Snort detection. We then analysed the collected traffic for entropy-based features to improve the detection accuracy of deep learning models: Stacked Auto Encoder (SAE) and Convolutional Neural Network (CNN). This work also investigated the trade-off between SAE and CNN classifiers by using accuracy and false-positive results. Quantitative results demonstrated SAE achieved relatively higher detection accuracy of 94% with only 6% of false-positive alerts, whereas the CNN classifier achieved an average accuracy of 93%.

Keywords: distributed denial of service (DDoS); entropy; software defined network (SDN); intrusion detection system

Citation: Ujjan, R.M.A.; Pervez, Z.; Dahal, K.; Khan, W.A.; Khattak, A.M.; Hayat, B. Entropy Based Features Distribution for Anti-DDoS Model in SDN. *Sustainability* **2021**, *13*, 1522. https://doi.org/10.3390/su13031522

Academic Editors: Tae-Eung Sung, Ki-Il Kim and Eungdo Kim
Received: 11 January 2021
Accepted: 25 January 2021
Published: 1 February 2021

1. Introduction

Digital services, such as banking, healthcare, education, entertainment, and national/local administration services to name a few, drive our modern society in which access to online services is often taken for granted. These services have become nonexclusive routines for almost everyone. Many of us check our official emails and social services first thing in the morning. The dependency of our day-to-day activities on these services introduces a large number of attacks on network services. The latest development in software, network, and system exploits and vulnerability tools has brought up new attack vectors to compromise access to an entire network or subnetwork. However, network defenders use up-to-date and the most sophisticated defence systems for their safeguard. Contrary to conventional host or service based attacks, Distributed Denial of Service (DDoS) attacks are considered more disruptive in nature. These attacks make targeted services unavailable by sending a significantly large number of malicious access requests to a service provider.

After resources being depleted, the service provider becomes unable to serve its potential legitimate users. Nowadays, DDoS is a commonly used attacking method which inflicts heavy financial and reputation losses [1].

With the advancements of virtualization-based computing, Software Defined Networking (SDN) has been widely adopted for security solution in various services and service provisioning models [2–4]. In SDN, most of routing and topological decisions are carried out by a separate entity called control-plane [5]. This decoupling approach has brought enormous benefits to network management and provides a feasible and effective solution to improve network efficiency [6]. Furthermore, the separation of data-plane and control-plane assists to manage a flexible and scaleable networking infrastructure to meet the day by day ever-changing modern business needs. Although the logical centralised architecture and its programability approach enables the SDN controller to detect malicious activities; however, the controller itself becomes vulnerable to DDoS attackers [7].

Most of the forwarding decisions are managed by the controller. The table in OpenFlow based switches consistently searches for new packets arrival, with a successful match, flow action is performed. If packet_in does not match then it is propagated to SDN main control-plane for detailed analysis. In case of DDoS attack, if the arrival rate of packet_in is significantly higher, then control-plane resources start to deplete, which results in discontinuity with data-plane and may overwhelm the controller. A single point failure, such as overwhelming the controller with malicious traffic could defunct the whole networking infrastructure [8].

Amongst existing security problems in SDN, DDoS attack is considered as one of the most urgent and hardest security issues [9]. So far, DDoS attack detection in SDN is well researched with approaches including [10–16]. However, most of these conventional studies are focused on attack detection and mitigation methods. Majority of work is based on time-based periodic detection—choosing a right time-period to detect an attack is very hard. If a large time-period is selected for attack launch then response time for detecting attack will be increased. This creates extremely large attack overload over deployed switches and its major SDN controller. In contrast, if the time threshold is set to a relatively low value, then deployed attack detection module will continuously run, which unnecessarily consumes controller resources, such as CPU, network up-stream and down-stream bandwidth. It also affects controller efficiency.

However, congestion at the controller is one of the major issues that could easily lower down the performance of deployed mechanism and easily left the entire infrastructure vulnerable especially for DDoS attacks. Currently, most of the research is not focused to improve the accuracy of the controller in SDN, as most of the detection modules work from the SDN controller. However, it is mandatory to rectify SDN controller efficiency with available characteristics of SDN. To solve the aforementioned issues, we propose a new-fashioned anti-DDoS detection mechanism with entropy-based feature distribution in SDN. Our detection model comprises of Snort alert based Features, Entropy Calculation, Feature Distribution and Traffic Processing, and Machine Learning Classifiers. We summarised our contribution as below:

- An effective anti-DDoS detection mechanism is proposed, to speed up major SDN controller unit accuracy so that deep learning model easily classifies trade-off between benign and unknown malicious code.
- We have distributed specific traffic features with generalised entropy estimation of Shannon and Renyi formulas.
- By utilising a Snort–Ryu implementation with entropy calculation, we acquired non-redundant traffic features.
- As detection classifiers, we utilise well known deep learning classifiers, such as Stacked Auto Encoder (SAE) and Convolutional Neural Network (CNN) to compare the accuracy and False-Positive alerts with 30% and 60% attack rate with normal traffic.

Rest of the paper is organised as follow: Sections 2 and 3 represent the related literature work and background in details, respectively. Our novel entropy based DDoS detection

model is provided with details in Section 4. Experimental results evaluations are presented in Section 5. The paper is concluded along with future directions in Section 6.

2. Related Work

In previous decades, most of the security research is performed mainly with legacy networks [17–19]. Different approaches have been implemented to detect and mitigate the DDoS traffic only in traditional network [20–23]. To meet the needs to digital services, traditional networks found computationally expensive, time-consuming and it requires more modern innovations for security implementation. OpenFlow enabled SDN infrastructure has been proved successful with various security challenges [14]. Although SDN provides a feasible and modern platform, the control-plane layer of SDN is not extensively researched from a security point of view. Due to the programmable model and logically centralised implementation, it brings new vulnerabilities and threats making SDN control-plane layer as an attractive target for potential intruders. Especially, massive DDoS attacks at control-plane of SDN based platform, which can result in the unavailability of the entire network [24].

One of the main challenges of network security and machine learning is to distribute and select optimal features [25]. The feature subsection aims to pick a feature subset that performs better within a certain condition of assessment [26]. According to the analysis of [25,27], feature selection can be classified from the technical viewpoint approaches. By adaptively validating the system with multiple combinations of features as inputs, also, existing researchers focused to collect and classify the ideal features for the optimal performance in proposed models.

To assess regular traffic flow, Mehdi et al. used the maximum entropy calculation methodology to address security challenges in SDN [11]. Experimental investigations was carried out by using OpenFlow NOX controller switches, using low-rate data traffic. Their main goal was to classify attack traffic in a home setting. In another research, investigators used the framework of entropy to predict the transmission of worms and threats through port scanning [13]. Besides this, Ref [10] suggested an entropy-based anomaly detection technique. In their proposed research, the identification module operated in the edge switches to lower down the overhead of the control plane. Kotani et al. suggested a packet filtering strategy to secure the controller [28]. Such a strategy identified the elements of the packet headers before the packet-in occurrence was forwarded. On the contrary, the strategy is ineffective unless the attacker produces new streams wherein the flows. Dong et al. developed a system for detecting the vulnerable applications for which attackers were connected [29]. Typically, the threshold was set in feature extraction, and any irregular variance of incoming traffic feature vectors helped to classify threat [30]. Mohammadi et al. suggested a prevention strategy to fight the TCP SYN DDoS attack targeting SDN. They used SDN's programmability [31] for identification purposes. The system, however, was also vulnerable to several other protocol attacks. DDoS attacks, as a comparison to other attacks, will cause a significant disruption of any sort of networking infrastructure [32].

Entropy is a common way of producing valuable traffic classification features and has been extensively seen in recent frameworks for DDoS attack detection [33]. Entropy is an analysis technique that scales the ambiguity about the content. In network activity, entropy uses a single value metric to identify the distributional variations in traffic [34]. This has been increasingly recognised that adequate analysis of such improvements can classify network anomalies [35]. A new study of identification showed that detection based on entropy has better detection efficiency than many other approaches [36]. The entropy-based features classification techniques are feasible and is widely used [37], which possess various significant features like effective and fast calculation, lower false alerts, and higher detection accuracy. In particular, entropy calculations are extended to input traffic attributes such as IP addresses of source and destination, the destination port of source and destination. For example, the high entropy value indicates that a significant

disparity occurs concerning entropy specified at source address and that the low entropy value indicates a reduction in the source of traffic packets. This is valuable for the detection system, as a standard DDoS attack with many attack sources of a single target typically has a high variation of the source address and a low variation of the destination address relative to a regular traffic.

Identifying DDoS malicious traffic at the data-plane layer is difficult because OpenFlow enabled devices have no self-adaptive intelligence to segregate network traffic flows. Addition to this, attackers use easily available tools and hardware-assets [38]. This section presents a systematic literature of DDoS detection solutions, which are widely deployed in SDN control-plane and listed in Table 1. Most of the existing approaches have evaluated DDoS detection techniques by classifying packet traffic either legitimate or malicious and broadly categorised into entropy-based anomaly detection, signature-based, machine learning-based and hybrid detection. These approaches are deployed in SDN infrastructure to detect DDoS traffic.

Table 1. The existing DDoS detection literature survey in SDN.

Author	Year	Description	Methods	Detection	Mitigation	Traffic Analysis
[12]	2010	Proposed work uses six SDN based traffic features for detecting DDoS.	Neural network model, SOM.	✓		✓
[11]	2011	This work utilises different SDN approaches to collect DDoS traffic.	Maximum entropy, TRW-CB, Rate-limiting .	✓		✓
[14]	2013	Proposed approach lower down the control-plane and data-plane overhead.	Interface migration technique.	✓	✓	✓
[13]	2014	Flow-based traffic utilised to classify DDoS portscan, and worm propagation.	Entropy-based detection.	✓	✓	✓
[39]	2015	Proposed method reduces requests burden by utilising scheduling technique in SDN.	Malicious traffic redirection approach.	✓		✓
[10]	2015	Model running on edge switch to detect DDoS with low control-plane burden.	Entropy-based Algorithm.	✓		✓
[40]	2016	This model utilises Shanon entropy estimationto detect early-stage DDoS attacks.	Attack sources bypass method.	✓	✓	✓
[29]	2016	Method of Sequential Probability Ratio Test (SPRT) used to classify high rate DDoS traffic.	SPRT method.	✓	✓	✓
[41]	2016	ML-based approach utilised with SDN flows to identify and mitigate the attack.	ML-SOM approach.	✓	✓	✓
[31]	2017	Proposed method is used to mitigate TCPflooding attack in SDN context.	TCP request monitoring, malicious hosts blocking.	✓	✓	
[42]	2017	multi-vector DDoS traffic analysis and detection system in deep learning	SAE, deep learning.	✓		✓
[43]	2018	Extreme gradient algorithm utilised to detect DDoS traffic with low false-alerts.	Machine learning model.	✓		

Some authors utilised entropy-based statistical techniques to analyse traffic [10,44,45]. The authors of [42,43] proposed an entropy-based technique to detect DDoS at POX controller during initial attack stage. This proposed work has a limitation, once the number of hosts increases the proposed model generates false positive alerts. The computational overhead from the controller is reduced by deploying fast-entropy approach with flow-based model [44]. The authors in [39] proposed a scheduling based method to detect DDoS, where a single processing queue is divided with subsets of *k* logical queues, each of them belongs to the network switch. During heavy traffic burst, the SDN controller utilises logical queues to satisfy scheduling request. The authors of [11] utilises maximum entropy estimation technique for classifying normal traffic distribution to solve home office network security concerns in SDN. Most of the experiments were deployed with OpenFlow enabled switches with NOX SDN controller. In [13], authors have used entropy methodologies for identification of port-scan attacks and worm propagation. Another entropy-based anomaly detection solution was proposed by the authors of [10], to detect DDoS attacks in SDN. This work more likely focused to reduce control-plane workload.

Some of other, well-known methodologies have been published to detect DDoS traffic with SDN based architecture, such as self-Organising Maps (SOM), which is Machine Learning (ML) based approach to detect malicious traffic [12]. This work uses only six features to classify the malicious attack traffic, i.e., Average Packet per flow (APf), Average Duration per flow (ADf), Average Byte per flow (ABf) etc. Similarly, Refs. [41,46], also utilised different ML-based approaches to classify traffic patterns. The authors in [16], proposed adaptive flow collection based DDoS detection model in SDN. This methodology utilises OpenSketch traffic measurement tool to create a hash table for measuring traffic. This approach uses three stages based pipeline process to gather traffic samples for identifying malicious traffic instead of using traffic flow sampling. In SDN, most of the DDoS detection solutions are carried out with the collaboration of ML and knowledge-based techniques to identify malicious attacks. Generally, ML-based techniques classify attack flows based on specific features. ML-based anomaly detection models are mainly suitable for small networks. For larger networks with heavy traffic flow overhead, are unmanageable for traffic collection and analysis inside controller [13]. Although during the attacking scenario, response time is more important to improve detection performance; however, ML performance is also dependant upon trained datasets and its features diversity.

Recently, ML-based detection techniques have been widely applied in SDN to address the challenges of DDoS detection. Authors in [42] proposed a semi-supervised one class-based Support Vector Machine (SVM) to classify anomalies, here the small quantity of malicious traffic is utilised as compared to normal traffic. This model is feasibly capable to detect outliers from the initial background traffic phase, which helps to easily manipulate majority of the traffic characteristics. The authors have used the Stacked Auto Encoder (SAE) to train datasets; however, it consumed a lot of time to process model iterations. Similarly, the authors of [43] proposed high precision DDoS detection model, which is based on Xboost classifier SDN. The proposed approach analysed most of the DDoS attacks to cater feasible and effective solution. In SDN, POX controller's grab bag connection, most of TCP, UDP, and ICMP flooding attacks were sent for manipulating connection records which enabled us to evaluate DDoS classifiers.

According to [47], a packet_in filtering approach can protect control-plane. This technique helps to list most of contents extracted from packet header field prior to sending the packet_in message. However, when intruders launch very distinctive flows in which all packets have different field values rather than specified values of the proposed technique, then it fails to capture malicious records. Authors in [29] deployed detection model for locating compromised interfaces, which are used by attackers during attack time. Most of anomaly detection model uses fixed threshold values, once incoming statistical features deviate with abnormal conditions it is identified as attack traffic.

From the literature survey, it can be seen that some research has been carried out for the detection of DDoS attacks by utilising traffic feature distribution with entropy-based

methodologies. By utilising feature distribution with the help of entropy calculations over existing detection techniques, primarily we can reduce redundant and unnecessary features processing overhead and improve the detection requiring relatively less time.

Convolutional or convolutional neural networks (CNN) [48] are known as enhancements of conventional feed forward networks (FFNs). These were initially tested for object recognition using Convolution 2D layers, 2D layer pooling and a totally interconnected layer. This was accompanied by the natural language analysis of the Convolution 1D layer, the pooling of 1D layer and the completely connected layer [48]. Whereas the conventional CNNs used mostly for image analysis with the help of 2D, 1D, as CNNs can be used effectively for time series processing, since time series in 1D can effectively derived by convolutions [49]. In our proposed study, we utilise the 1D CNN as deep learning classifier to identify security threats in complex multivariate and distributed features based on entropy estimation.

In CNN, convolution is used as primary building block, where entropy based input features converted as 1D time series input vector of $z = (z_1, z_2, z_3, \ldots, z_n)$. All distributed features based on entropy calculation are fed towards the fully connected layer of CNN, a fully connected layer comprises on the *soft-max* function, which actively utilises the probability distribution with input features vector one by one. CNN layer with fully interconnected *soft-max* function is provided as below in Equation (1).

$$\theta_z(Z) = softMax(\frac{e^z}{\sum_{i=1}^{n} hl + b_0}) \quad (1)$$

where hl utilises the highest feature value connected to each input vector of $z = (z_1, z_2, z_3, \ldots, z_n)$, and b_0 is used for non linear activation function.

The SAE consisted of several self-encoders—input or visible layer, a hidden layer, and a output layer also called reconstruction layer. The input data is loaded into the visible layer. The construction layer is inducing output. The SAE architecture is special in design relative to CNN, DBN, and RBM deep learning models. In the first place, SAE is made up of a basic and straightforward structure and is trained in a much shorter time compared to the other described Deep Neural Network (DNN) algorithms [50]. Second, because of the nature of the unsupervised learning strategy, SAE is not using labelled datasets. On the other hand, CNN is based on supervised learning, while DBN and RBM use supervised learning. Finally, the SAE algorithm employs outputs as inputs, and detailed features components can be retrieved with a useful training strategy in the SAE. This paper uses comprehensive features of an SAE method based dataset to increase the rate of identification of DDoS attacks in SDN. SAE as DNN uses sparse auto-encoders and soft-max classifiers to extract and label unattended data.

In Equation (2), β values denotes sparsity penalty for the weight coefficient via Kullback-Leibler divergence. This divergence function enables to input features vector such as $z = (z_1, z_2, z_3, \ldots, z_n)$ to process if there is a possibility of lower average activation function, when $\rho = \rho_J^{\wedge}$ then this function comprises the minimum values of 0. During training stages of input layer values, $\rho_J^{\wedge}$ is utilised for an average activation with Jth values and ρ is used for sparsity coefficient at hidden layer.

3. Detection Methodology in SDN

This section presents the proposed DDoS attacks detection methodology and its implementation details. The detection of the model relies on the specific DDoS feature distribution approach, which is achieved by generalised entropy (GE) calculation with Shannon and Renyi formulas, details of entropy distribution is provided in this section. Our detection methodology comprises: data acquisition with Snort–Ryu, feature distribution with entropy calculation, data processing and traffic classification with SAE and CNN. Overall, the general overview of our proposed DDoS detection methodology presented in Figures 1 and 2 elaborates the implementation detail.

$$J_{sparse}(z) = J(z) + \beta \sum_{J=1}^{q} d_{KL}(\rho \in \rho_J^{\wedge}) \quad (2)$$

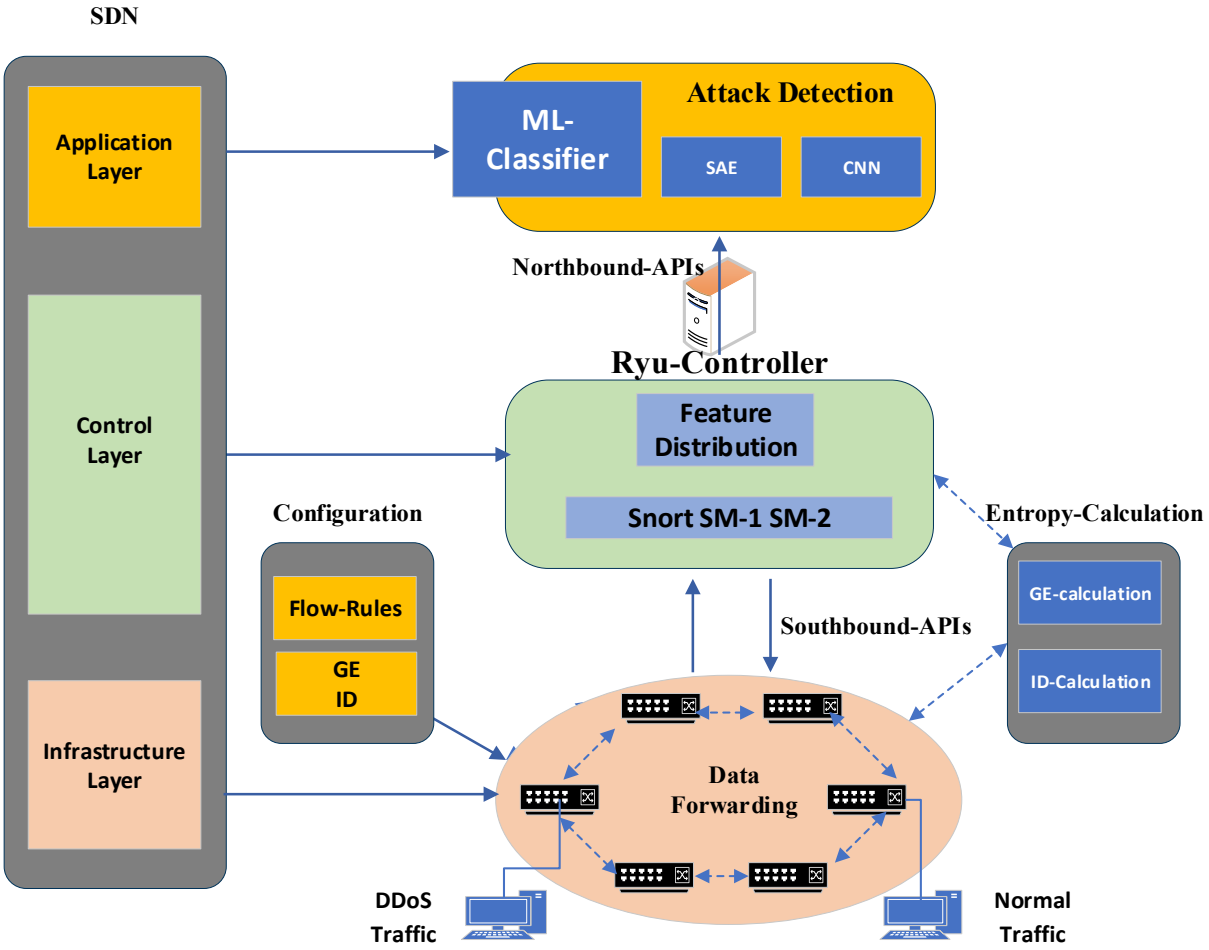

Figure 1. General overview of proposed system in SDN.

3.1. Snort–Ryu Based Data Acquisition

This section comprises on the huge amount of live datasets acquisition from the network, due to heavy network traffic human inspection and data analysis is unmanageable. The Snort [51] is capable of being used as a signature-based detection engine or as a log_tcpdump module with various output files. Although, log_tcpdump can be utilised for storing test-bed datasets, which is limited to store only 128 MB of total packets. We extended storage module of Snort by implementing a Barnyard2 logfile. We have utilised Snort–Ryu modular implementation to collect the test-bed datasets. The Snort [51] engine consists of various traffic attributes, such as *timestamp, sig_generator, sig_id, sig_rev, msg, proto, src, srcport, dst, dstport, ethsrc, ethdst, ethlen, tcpflags, tcpseq, tcpack, tcplen, tcpwindow, ttl, tos, id, dgmlen, iplen, icmptype, icmpcode, icmpid, icmpseq*. Our proposed DDoS detection model based on feature distribution is dependant on the features, such as time window, protocol, source IP, address, source port address, destination IP address, destination datagram length, port address, priority, etc. By utilising Snort, our model acquires relevant features in collaboration of entropy calculation. Snort utilises two different modes to capture distinct features for malicious and benign network traffic. SM-1 and SM-2 modes depicted in Table 2 are used to collect malicious traffic and benign traffic features.

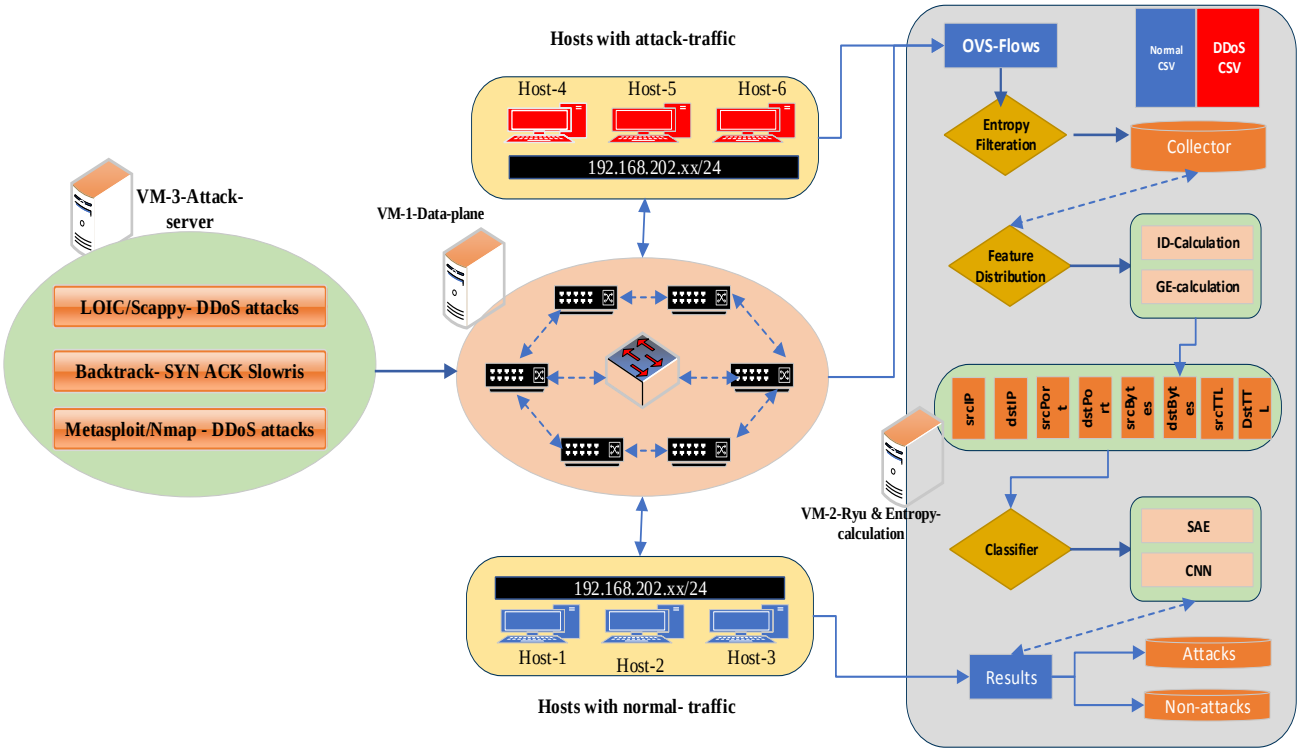

Figure 2. Proposed system of features distribution in SDN–Ryu-controller.

Table 2. Two modes of Snort.

Description	Snort Two Modes	
	SM1 (Malicious)	SM2 (Benign)
TCP rules	20 signatures	8 signatures
UDP rules	10 signatures	6 signatures
ICMP rules	6 signatures	6 signatures

3.2. Entropy Calculation

The proposed methodology is depicted in Figure 2. The main aim of our model is to capture various types of DDoS attacks, for this purpose, the proposed model mainly relies on the finding most common attributes from the flows. One of the most common attributes is source IP addresses which generate attacks. Once various types of features are collected with SM-1 and SM-2 modes, the proposed model takes advantage of Shannon entropy estimation to generalise most relevant types of features that lead to successful DDoS attacks detection in SDN. This model utilises $H_\alpha(Z)$ Shannon entropy formula to identify relevant features from Snort model.

Addition to this least values of entropy is estimated in case of small uncertainty. The event distribution randomness is evaluated with Renyi entropy metric order with α, this enables Shannon entropy estimation towards more relevant features generalisation. The Generalised Entropy (GE) of discrete variables Z with possible number of outcomes, such as, $z_1, z_2, \ldots, z_n$, which can be accumulated, i.e., $\sum_{i=1}^{N}, 0 \leq Z_i \leq 1$, then entropy of Renyi with order α can be defined as below:

$$H_\alpha(Z) = \frac{1}{1-\alpha} \log_2 \left(\sum_{i=1}^{N} p_i(\alpha) \right) \tag{3}$$

Here values of $\alpha \geq 0$ and $Z_i \geq 0$. Different entropy calculation is performed in order to quantify various α orders. With substitute of $\alpha = 0$, we get maximum generated information and with substitute of $\alpha = 1$ we achieve GE values, which is given below:

$$H_1(Z) = \sum_{i=1}^{N} p(z_i) \log_2 \frac{1}{p(z_i)} = -\sum_{i=1}^{N} p(z_i) \log_2 p(z_i) \quad (4)$$

Here, $p(z_i)$ = probability $(Z = z_i)$ is the ith value of Z. The Equation (4) is known as Shannon Entropy, when we put $\alpha = 2$, then GE expression is depicted as:

$$H_2(Z) = -\log_2 \sum_{i=1}^{N} p_{i=1}^{\alpha=2} \quad (5)$$

Equation (5) is known as Renyi entropy estimation, the authors of [52] have depicted relationship between Renyi and Shannon, where GE estimation values relies on α values, such as $\alpha = 1$ or $\alpha = 2$. GE values exponentially increase with various probability distribution as compared to Shannon entropy probability [53,54]. Following this work, we classify our traffic as a benign and malicious probability distribution. To get GE values, our work also manipulates both different probability distribution values with a combination of attack and benign properties. Once we get higher uncertainty events then probability results in more GE information as compared to Shannon entropy [55].

By adjusting α values in GE, we can get different values of entropy to meet our DDoS detection methodology. This paper utilises Information Distance (ID) with the help of Renyi and Shannon GE estimation values via $\alpha = 1$ or $\alpha = 2$. this helps to estimate event's similarities—the methodology is depicted with two different probability distribution such as $P_{ID} = \{p_1, p_2, \ldots, p_n\}$ and $Q_{ID} = \{q_1, q_2, \ldots, q_n\}$ as below:

$$\sum_{i=1}^{N} p_{i=1} = \sum_{i=1}^{N} q_{i=1} = [1 - 0] \quad (6)$$

The ID equation can be derived as below:

$$D_\alpha(P_{ID}, Q_{ID}) = \frac{1}{1-\alpha} \log_2 \left(\sum_{id=1}^{N} p_{id}^{\alpha} q_{id}^{1-\alpha} \right) \quad (7)$$

ID always focuses non negative values, such as $\alpha \geq 0$. However, if both probability distribution are similar then $D_\alpha(P_{ID}, Q_{ID}) = 0$, GE entropy expression can be achieved by varying α orders as given below:

$$D_1(P_{ID}, Q_{ID}) = -\log_2 \left(\sum_{id=1}^{N} p_{id} \right), \Rightarrow \alpha = 1 \quad (8)$$

$$D_2(P_{ID}, Q_{ID}) = -\sum_{id=1}^{N} p_{id} \log_2 \left(\frac{p_{id}}{q_{id}} \right), \Rightarrow \alpha = 2 \quad (9)$$

The Equation (9) is known as Kullback–Leibler divergence (KL) distance. This equation is utilised for measuring ID with identical, triangular inequalities and symmetrical properties of KL divergence. However, GE and ID both use these properties for DDoS detection to rectify DDoS most relevant traffic features with the help of Equations (5) and (7). In our approach, probability distribution is calculated by $H_\alpha(Z)$ as shown in Equation (3), where z_i depicts packets header variation between source and destination communication junctions that comprises of *Src-IP, Src-Port, Dest-IP, Dest-Port, Source-Bytes, Destination-Bytes, TTL, Flags, proto, Distinct Datagrams*. In Figure 3, a flow diagram is provided for the distribution of features. This is achieved by combining Shannon entropy and Renyi entropy formula. Our proposed work utilises probability distribution measuring approach to generalise relevant features from DDoS and malicious packet header. We manipulated all incoming packets with GE and ID metric to formulate our SDN based test-bed network to effectively detect DDoS traffic. Major aim focuses to lower down redundant features with GE and ID (information distance) entropy estimation such as:

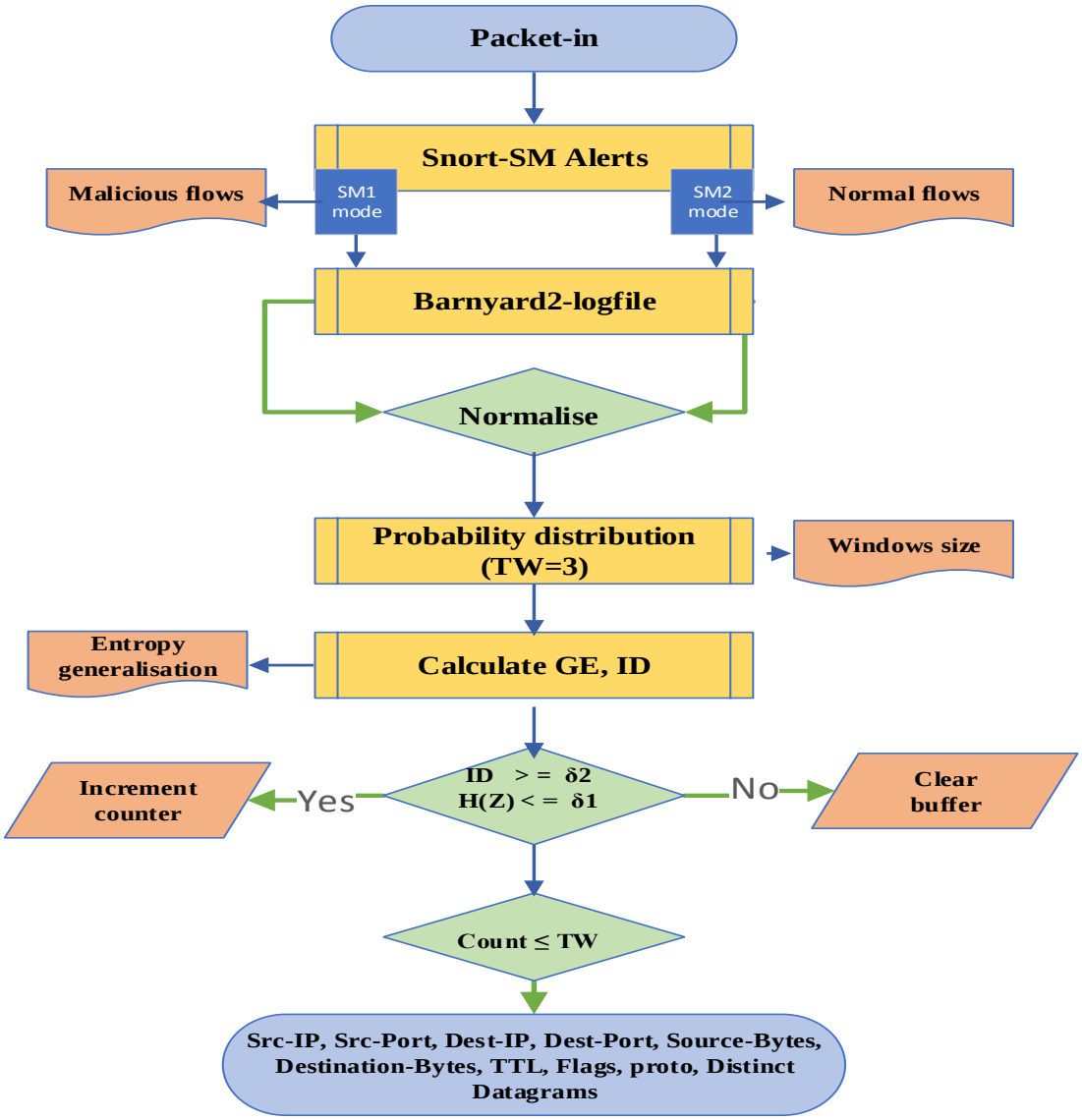

Figure 3. Flowchart of features distribution in an SDN–Ryu-controller.

3.3. Features Distribution and Traffic Processing

This section presents features selection and processing of datasets for our effective DDoS detection model. Major focus of selecting features relies on GE and ID entropy estimation to qualify relevant features as discussed before. However, an effective DDoS classifier needs many important kinds of traffic features distribution. This kind of traffic anomalies randomly changes during the distribution of most feasible addresses, ports, data length etc., in an observed traffic area. The overview of selected traffic features is presented in Table 3, which is achieved by distributing specific feature from test-bed datasets with the help of Shannon and Renyi joint entropy.

In DDoS detection event, the packet flowing rate is significantly higher than benign traffic, subsequently, packet diversity is also changing to generate entropy. In real-time traffic which is a complex form of different data rate, which remains stable with packet diversity and it also results irrelevant data pattern flows due to different traffic services. This is why entropy values are exponentially changed with respect to time. Due to the fact of such variation, setting a proper threshold to detect DDoS in the network is unmanageable.

We developed dataset to evaluate our proposed method. Our dataset is a combination of real-time legitimate data and synthesised malicious traffic, which is emulated with DDoS

attacking tools comprises of Metasploit, Scapy, HPing, and Low Orbit Ionic Cannoin (LOIC). We run Scapy and HPing DDoS scripts with various attributes of DDoS from the remote virtual machine as shown in Figure 2. For the validation purpose, we simulate various DDoS attacks on widely targeted ports. Most important realistic network traces are less publicly available due to privacy concerns and to label properly which require some manual entries. With the help of deep domain knowledge and feasible tools and methodology, this approach enables to create realistic dataset. Our proposed model calculates and classify benign and malicious traffic using entropy estimation followed by two major components as window size and two threshold values. For window size, we utilised number of packet received. For entropy values we estimated the incoming packets occurrences with windows size, such as, if number of frequencies for each (dst_{IP}) is equably distributed then maximum value of entropy is established. If there is a sudden deviation, such as rapid decrements in entropy values for the same network, a malicious traffic flow event may occur.

Table 3. Selected traffic features with GE.

Features	Description
Src-IP (src_{IP})	Source IP address
Src-Port (src_{Port})	Source Port
Dest-IP (dst_{IP})	Destination IP address
Dest-Port (dst_{Port})	Destination Port
Source-Bytes (dst_{Bytes})	Total Bytes from source
Destination-Bytes (dst_{Bytes})	Total Bytes to destination
TTL Flow-Duration (TTL_{FD})	Time To Live duration
Flags	TCP, UDP, HTTP, ICMP flags
Proto (pro_t)	Types of variosu protocol
Distinct Datagrams $(Datagrams_{\Delta})$	Various Datagrams values

Similarly, main function of Algorithm 1 is to calculate the average entropy values for (GE, ID) for input features, which are presented in Table 3. SDN programability approach via collaboration of GE and ID also help to rectify specific features with more malicious attributes. Algorithm 1 helps to compute average entropy values (GE, ID), referred as AverageEntropy (GE, ID) in Algorithm 1.

In our proposed Algorithm 1, AverageEntropy (GE, ID) is calculated with two thresholds, such as lower δ_1 and upper δ_2 for every input stream with different time. Our aim is to classify and distribute specific features from incoming network stream. The distributed features are processed with the help of two different time slots, which comprise of deviation beyond normal ranges of δ_1 and δ_2. Our model uses entropy deviation such as a sudden rise of values or a drop of values as compared to predefined threshold values between 0 and 1. For example, during the event of DDoS attacks such as port scan on a specific location will result in dispersion known as entropy. We utilised two threshold values, lower δ_1, which is used for GE entropy values calculation with, and upper δ_2 is also used for ID values calculation. GE and ID are calculated for benign and malicious traffic with the help of Equations (4) and (7), such as if $ID \geq \delta_2$ or $H_\alpha(Z) \leq \delta_1$.

In Algorithm 1, entropy estimation is presented which works with time-based sliding. The Algorithm 1 initialises with time windows size setting, which effectively maintains resources utilisation and network flow deviations. We used only three minutes time slot, as using larger time needed more resources for processing and storage. The Algorithm 1 produced average output with estimated entropy of feature set S, which used only two threshold, such as Lower δ_1 and upper δ_2 for every S feature order. Major aim was to focus to classify and maintain very specific features from incoming network.

Algorithm 1 Entropy-based feature distribution

Input: Traffic data, Feature set S, δ_1 δ_2
Ensure Output: AverageEntropy GE, ID hash-table H_{tb} (src_{IP}), (src_{Port}), (dst_{IP}), (dst_{Port}), (src_{Bytes}), (dst_{Bytes}) , (src_{TTL}), (dst_{TTL})
TW $\longleftarrow$ 3 min
$H_{tab} \longleftarrow \{S\}$
for all IF $\in H_{tb}$ **do return** True find GE, ID of both flows with Equation (3)
if incoming packets $\in$ TW **do** search in S
else
if $ID \geq \delta_2$ *and* $Z_\alpha(IF_{at})$ *or* $Z_\alpha(IF_{be}) \leq \delta_1$ then do;
Utilise $p(z_i) \longleftarrow \left(\sum_i^N z_i\right)/N$ Estimating mass function probability
Utilise $H_i(z) \longleftarrow H(z)/\log n$ Entropy normalisation
Utilise $H_{tab} \longleftarrow H_{tab} \cup H_i(z)$
end
AverageEntropy (GE, ID) $\longleftarrow$ sum $H_{tab}/4$
end

The rectified features from network connection were processed with the help of two feasible tasks of time slots, which had AverageEntropy values beyond normal ranges of δ_1, δ_2. First of all, each abnormal subset of network traffic time slot was cleared by adding empty values rows to maintain a stable and unique column and row-based format. Secondly, cleared subsets were normalised by using *MinMax* function. The *MinMax* function generated feature values between scale ranges of 0 and 1. The *MinMax* feature normalisation function is depicted below:

$$Z_i^{nor} = \frac{z_i - min(Z)}{max(Z) - min(Z)} \tag{10}$$

where Z represents subsets of malicious traffic features and z_i represent current values of Z to make normalisation. The term Z_i^{nor} is called final normalised features within the ranges of 0 and 1.

4. Experimental Setup

Our research experiment is conducted with three different virtual machines on a workstation by using Intel Xeon X5560 CPU with 2.88 GHz processor and 16 GB RAM (DDR3 ECC-Registered Memory PC3-12800MHZ). We run TensorFlow 1.4V and Mininet on Ubuntu LTS 16.04-64 bit operating system. The proposed functionality is illustrated in Figure 2. We use VMware player to create virtual machines: VM1 with 192.168.202.x1 IP address, VM2 with 192.168.202.x2 IP address, and VM3 with 192.168.202.x3 IP address.

In our proposed model, a python script is used to generate attack and benign traffic with the Scapy tool. We also generated benign traffic with the help of normal web searches and browsing, and video streaming to validate the benign traffic of Scapy. We limited our model bandwidth up to 50 MB for 5 min interval to evaluate the performance, where TCP stood at 9 MB, UDP at 32 MB, and ICMP nearly at 2 MB, TCP, UDP, and ICMP minimum range were 380 Kbit/s and maximum range stood at 700 Kbit/s during attacks.

In VM1, Mininet as a the network emulator is utilised for creating 6 hosts with 6 OpenFlow switches. Kernel name-space properties of Mininet enables to prototype overall network environment within a single workstation. In Mininet, each process has its own network interfaces and routing table, these features enable to virtualise all network elements in Kernel. We connect these switches with Ryu controller with the help of OpenFlow (OF) version 1.3.

In VM2, the Ryu controller is implemented with Snort-IDS as Network Intrusion Detection System (NIDS) and entropy algorithm, which is presented in Section 3. This VMs play a vital role to collect networks traffic, then apply entropy probabilities property for feature distribution. The Snort-IDS collects every *incoming_packet* in Barnayrd2 log file, then Ryu based GE and ID entropy estimation reduces redundant features from all

network traces. Our detection model relies on more specific features, which are collected with GE and ID feature distribution for the deep learning classifiers. SDN controller is deployed in VM2, which centrally handles all virtual machines of our test-bed. Network policies are installed via REST APIs. In our system, VM1 considered as data-plane and VM2 as a control-plane. *ovs-ofctl* utility is used to insert network policies in the switches table, addition to this, the ovs-ofctl utility is also utilised for monitoring and administration purposes between data-plane and control-plane.

In VM2, Ryu uses two network interfaces, one in promiscuous mode on *eth0* interface to collect all OpenFlow traffic traces with Snort from VM1, while another *Eth1* is utilised as a port mirror for entropy calculation on Snort-Barnyard2 packets. Snort as NIDS plays a very vital role to acquire all raw network traffic from our proposed model. Snort switch $(switch\_snort.py)$ application is implemented on the top layer of Ryu controller, which helps to support Layer L2 switch code and also redirects feasible traffic by using OpenFlow enabled promiscuous mode. The Ryu controller receives Snort alerts by utilising $unixsock = false$, which helps to collect network packets and then store log-file in Barnayard2. This helps to manipulate data-plane traffic.

VM3 generates malicious traffic remotely, as illustrated in Figure 2. It utilises Scapy, LOIC, and Metasploit DDoS penetration testing frameworks such as Network Mapper (NMap) and Nessus Vulnerability Scanner. Scapy is considered powerful and feasible to launch real flooding attacks; however, our proposed work uses Scapy and LOIC to launch various TCP, UDP, and ICMP traffice. To validate our proposed work, we perform an attack and normal traffic on same VM1 data-plane area which is directly connected on the main SDN controller with deployed parameters. A python script is used to generate attack traffic and benign traffic with Scapy, where we select different hosts and source nodes during all injection. Our work also generates benign traffic by web activities such as web searches and browsing, and video streaming.

The probability of GE and ID entropy is applied on all test-bed datasets to acquire more special features set with no redundant features attributes. These features are which are provided in Table 3. Tshark and Tcpreplay tools are utilised to manipulate and analyse benign and malicious traffic individually. Once malicious traffic traces are classified with GE and ID as discussed in Section 4, then we categorise normal and benign CSV files into training and the testing datasets shown in Table 4. All datasets have values of non-zeros numbers due to unity based *MinMax* normalisation. We break our proposed CSV datasets into *0s* and *1s* values, normalisation as defined in Equation (10). After normalisation we utilise SAE and CNN deep neural network models to classify as an attack and non-attack values.

Table 4. Representation of normal and attack records for training and testing.

CSV 1	**30% Attack Rate Traffic**	
	Training	**Testing**
Normal	94,330	22,016
$All_{attacks}$	67,794	17,348
CSV 2	**60% Attack Rate Traffic**	
	Training	**Testing**
Normal	51,730	16,518
$All_{attacks}$	113,350	27,750

Performance Evaluation

In this work Snort IDS is also used for collecting all data-plane traffic from test-bed. Snort with two different modes: SM1 mode to acquire only malicious traffic, and SM2 mode for acquiring benign traffic. These two modes are configured with specific signature rules of Snort detection engine as illustrated in Table 2. We process Snort alerts

for feature distribution and generalisation by using GE and ID matric to analyse and calculate *Src-IP, Src-Port, Dest-IP, Dest-Port, Source-Bytes, Destination-Bytes, TTL, Flags, proto, Distinct Datagrams*.

In order to validate the effectiveness of detection model with proposed GE and ID feature distribution on the SDN controller, in proposed work, we are using two different scenarios. In the first one, we have used different attacking intensity, which is launched from a single host but remotely connected with Mininet data-plane VM1. We randomly generate attacks by using Scapy, LOIC, and Metasploit. The first scenario uses 20%, 30% and 40% attack rate. The second scenario uses 60% and 70% malicious traffic. In both scenarios, we generate benign traffic by using normal web searches and browsing, and video streaming. These searches are performed within VM1, where test-bed data-plane is created with Mininet. We maintain the attack intensity by using the following percentage equation:

$$Attack_{intensity} = \frac{Z_{attack}}{Z_{total}} \times 100 \tag{11}$$

In this equation, Z_{attack} represents the attack packets and Z_{total} represents total number of packets flowing in our test-bed. We run our code 10 times in each case for setting threshold values. We find the False-positive (FP) rate decrements, but the False-negative (FN) rate is stable. Table 5 represents threshold values during different attacks scenario.

Table 5. Parameters of proposed model with different attack rates and average threshold values.

Parameters	20% Attack Traffic	30% Attack Traffic	60% Attack Traffic	70% Attack Traffic
Mean-value	0.7862	0.7388	0.6475	0.3426
St-deviation	0.0249	0.0289	0.0381	0.0389
Max-confidence	0.7806	0.7438	0.6518	0.3815
Min-confidence	0.7899	0.7318	0.6437	0.3699
AverageEntropy (δ1, δ2)	0.7901	0.7636	0.6478	0.3910

Our model mainly relies on average output with input data (estimated entropy values)—it uses two thresholds, such as lower δ_1 and upper δ_2, for every input data collected based on different time. Our aim is to classify and distribute specific features from incoming network stream. The distributed features are processed with the help of two different time slots, which comprise of deviation beyond normal ranges of δ_1, δ_2. Our model uses entropy deviation such as a sudden rise of values or a drop of values as compared to predefined threshold values between 0 and 1. For example, during the event of DDoS attacks such as port scan on a specific location will result in dispersion known as entropy. Following steps are carried out to set up threshold values:

- We calculate possible maximum attack traffic values, this is achieved by combining attack traffic mean entropy values and confidence interval values.
- After taking the difference between these values, we derive δ_2 values for mean and standard deviation.

In Figure 4, we utilise 30% and 40% attack rate, while Figure 5 uses 60% to 70% attack rate. Each points on horizontal line represent windows size and vertical line represents entropy values, such as E_{values}. In Figure 4, blue curve represents the normal traffic and orange curve represents malicious traffic. We stabilise network by injecting manipulated malicious traffic remotely and run Algorithm 1 10 times. However, benign traffic entropy values is common in all attack rates as shows in Figures 4 and 5.

We have considered different attack scenarios with a single victim and multiple victims. In single attack victim only single host is under attack. On the other hand multiple victims attacks, we have launched attacks on 5 hosts. During simulation, the deviation and sudden drop of entropy values are considered to be used for traffic feature distribution. The rapid drop into flows represents malicious activity based on this phenomenon.

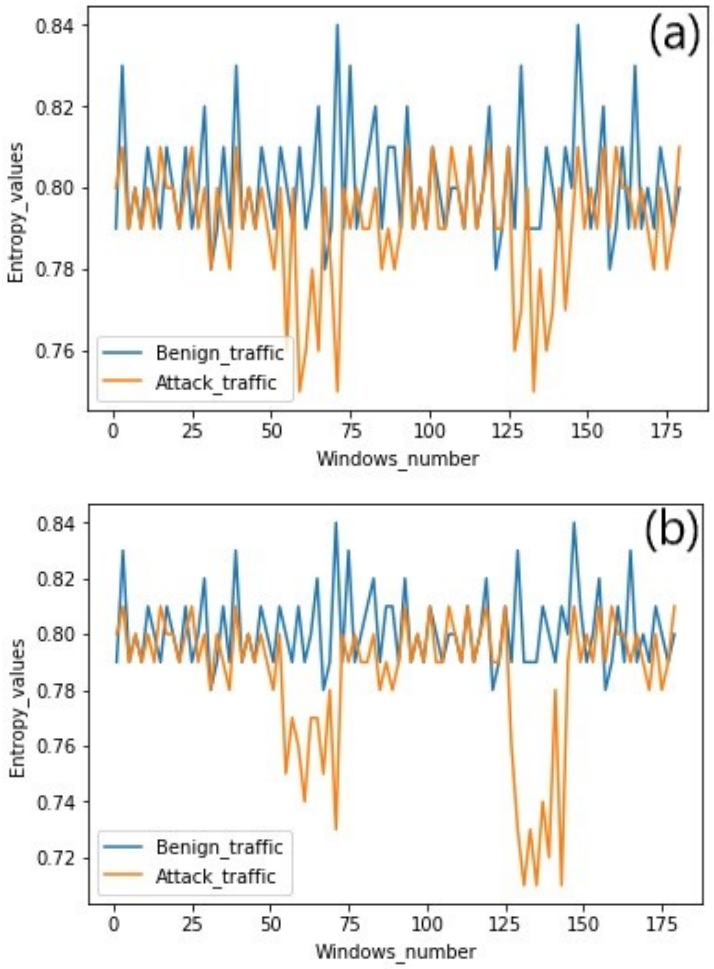

Figure 4. Entropy variation with attack rates. (**a**) 30% Attack Traffic. (**b**) 40% Attack Traffic.

From Figure 4, it can be seen that entropy values drop is least significant in case 30%, 40% of attack rate. However with 30% attack traffic, the mean entropy values are found as 0.77 on 53 windows interval, 0.72 on 57 windows interval, 0.76 on 63 windows interval, 0.72 on 71 windows interval. After every 25 windows intervals, entropy values dropped at average values of 0.72 to 0.77. With 40% attack rate, we found mean-values of entropy drops from 0.82 to 0.74 on windows interval of 55, after every 25 consecutive intervals, mean values repeatedly fall between 0.73 and 0.71.

Moreover, there is significant entropy values change in the case of 60%, 70% of the attack rate. In Figure 5a, the mean value of entropy drops to 0.64% after every 25 windows intervals. Similarly, Figure 5b, the mean value of entropy exponentially drops to 0.35% on 55 windows interval after 25 consecutive windows intervals mean values constantly falls to 0.35%. As compared to benign traffic mean values, attack traffic mean values drop around 0.50%, which is significantly higher with 70% attack rate through all experiments. This entropy value is far less than the threshold values, which is very feasible to classify this event as malicious.

Benign traffic is common in all attacks experiments, it is fixed threshold values to compare entropy values deviations. In Figures 4 and 5, entropy deviation values are less than the fixed threshold or some times it is higher than the fixed threshold. However, we have acquired traffic and classified based on mean values, which are significantly less than a fixed threshold as already illustrated in Figures 4 and 5.

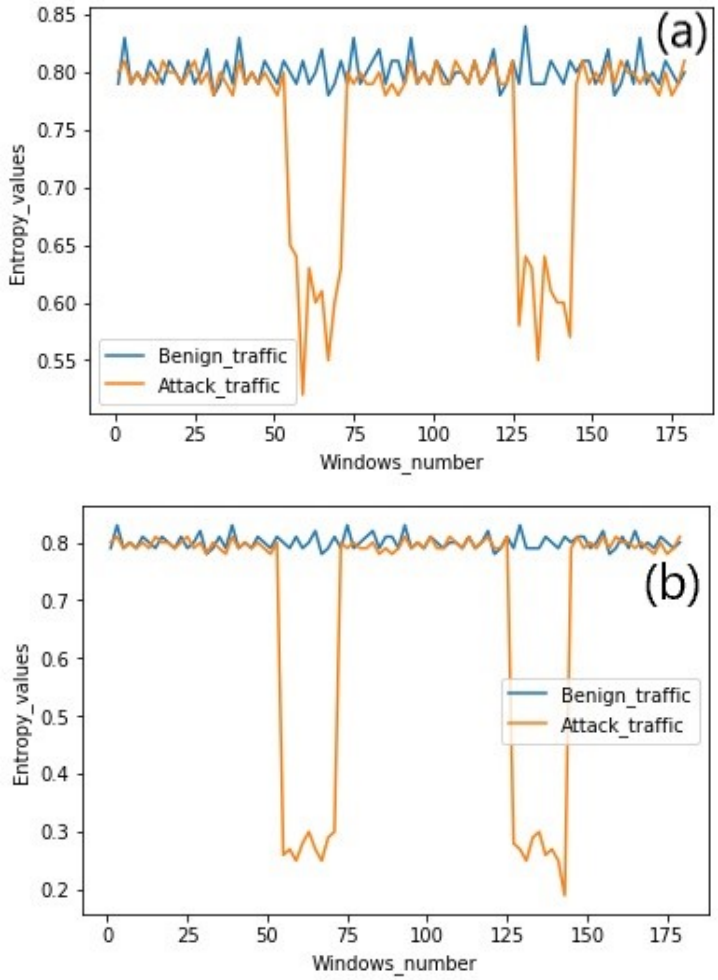

Figure 5. Entropy variation with attack rates. (**a**) 60% Attack Traffic. (**b**) 70% Attack Traffic.

5. Results

The major objective of our proposed work is to improve the accuracy of the deep learning-based model to detect various DDoS attack traffic, for which we mainly rely on malicious traffic features distribution and manipulation. For demonstration purposes, we used Snort as a NIDS, the practical application of this work is independent of any specific type of NIDS.

We have utilised collaborative approaches of Shannon and Renyi entropy such as GE and ID with SDN based programmability. This approach is able to lower down the unnecessary and redundant features from malicious traffic, which enable classifier to improve detection accuracy rate. Similar work has already been discussed in [40], the authors utilised Shannon entropy calculation as detection metric in their proposed work. However, in our contribution, we have utilised combined entropy calculation with Shannon and Renyi formulas to generalise DDoS traces from test-bed with traffic distribution approach.

Based on the various parameters such as Table 3, we classify our specific malicious traffic features comprise of *Src-IP, Src-Port, Dest-IP, Dest-Port, Source-Bytes, Destination-Bytes, TTL, Flags, proto, Distinct Datagrams*. We carry out our experiments based on the above design, and utilise four different attack rates with normal traffic also depicted in Table 5, the window size is fixed to 180 seconds for attacking and normal hosts used in our network.

5.1. Selecting DDoS Classifier

In order to get feasible detection results, we selected two SAE and CNN. First, we acquired input vector based on entropy feature distribution and then compared SAE and CNN detection accuracy and False Positive (FP) rates based on collected datasets (Table 3).

Our work uses different malicious rate such as 30% and 60% of attacks intensity with benign traffic, which results in a higher level of unbalanced features. In such cases, SAE and CNN detection algorithms fail to classify attacks. To overcome this issue, the authors of [56] have improved higher accuracy of detection model by using weighted loss function to stabilise the test-bed features. Our proposed SAE and CNN detection model uses well known metrics such as *precision, recall and F1 score* on datasets specified in Table 3. These metrics are very useful for the goodness of the detection rate of the proposed model. With the help of the confusion matrix we have provided parameters and its entries as below:

- True Positive (TP)—Identifying values which are correctly identified as attacks records.
- True Negative (TN)—Identifying values which are correctly identified as non attack values.
- False Positive (FP)—Identifying values which are incorrectly predicted as attack records.
- False Negative (FN)—Identifying values which are incorrectly identified as non attacks

1. Precision (P): To identify proportion of predicted real attacks with correct values:

$$P = \frac{TP}{TP + FP} \tag{12}$$

2. Recall (R): To calculate percentage of predicted attacks with all available attacks. However, higher value of R is very important:

$$R = \frac{TP}{TP + FN} \tag{13}$$

3. F-measure (F): To calculate model accuracy by utilising harmonic mean with both of precision (P) and the recall (R) values, higher F-value is considered feasibly important:

$$F = \frac{2}{\frac{1}{P+R}} \tag{14}$$

Our work uses two types of CSV files with different malicious and benign traffic. In the first one, we combined 30% attack rate with normal traffic and the second one utilised 60% of attack rate with normal traffic as shown in Table 4. These CSV files are also divided by training and testing section followed by 80% and 20% rule of training and testing, respectively. We adjusted the SAE detection algorithm with only three hidden layers and these three hidden layers are set as descending order with the following hyper-parameters which are depicted in Table 6.

Table 6. Proposed model configuration values.

Hyper Parameters	Model Values
Total hidden-layers	1, 2, 3
Hidden-layers size	6, 3, 2
Learning-rate	0.1, 0.01, 0.001
Activation-function	Sigmoid
Dropout	0.75, 0.50
Batch-size	100, 50, 50

In this paper, we have compared performance metrics by using SAE and CNN detection classifiers. We compare the SAE and CNN separately with the same traffic consisting of TCP, UDP, ICMP packets depicted in Figures 6 and 7. The first case uses 30% attack combined with normal traffic and the second case, utilises only 60% attack traffic with normal for better evaluation purpose.

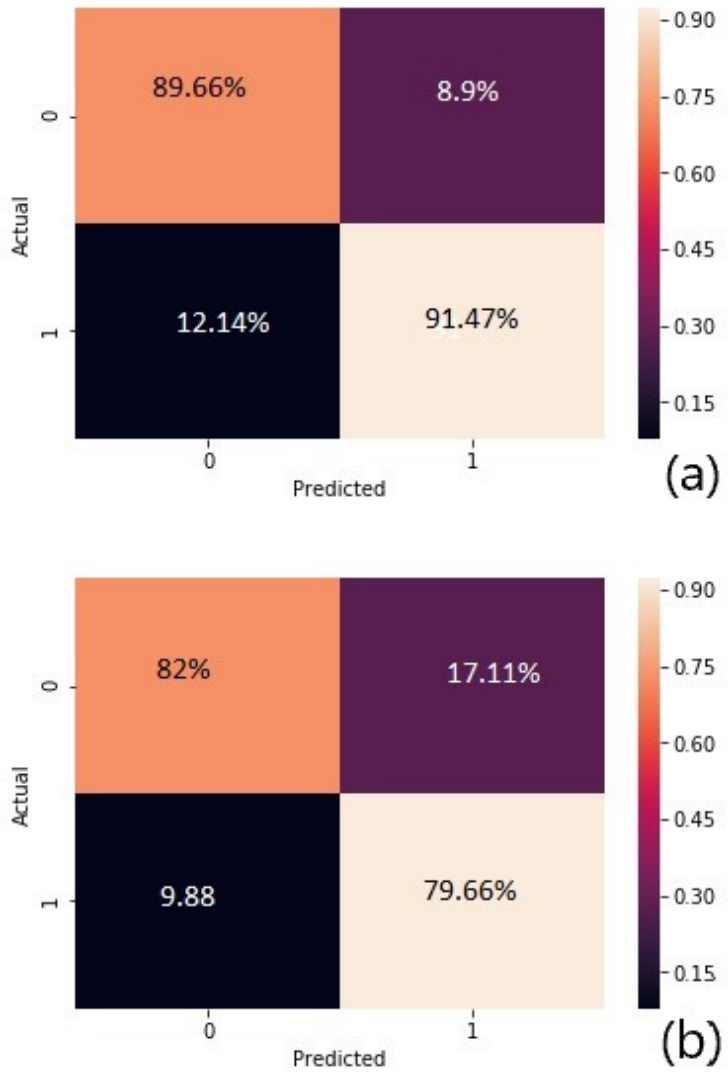

Figure 6. SAE and CNN Confusion matrix with 30% Attack Traffic. (**a**) Confusion matrix of the SAE Algorithm. (**b**) Confusion matrix of the CNN Algorithm.

Figure 6 represents the confusion matrix for SAE and CNN detection model for 30% attack rate. The Figure 6a confusion matrix of SAE depicts 91.47% of TP detection rate with less than 9% of FP rate. SAE model achieves more than 12% of FN rate with first case traffic. Similarly, as shown in Figure 6b, the CNN model is not receiving higher correct detection as compared to SAE. The CNN detection model with 30% attack traffic can only classify only 80% and 82% of TP rate and TN rate, respectively, the FP rate is around 17% of false triggers.

Comparing performance metrics of SAE and CNN with confusion matrix with 30% attack, SAE achieves 82.4% accuracy; whereas, CNN detection model achieves only 76.52% accuracy. Although SAE detection model is lightweight with only three weighted hidden layers, which require time for training and testing as compared to CNN model.

In Figure 7, comparison of SAE and CNN confusion matrix is presented with 60% of attack rate, where Figure 7a shows 97% and 91.22% of TP rate and TN rate, respectively for SAE. The FP rate is also acceptable, which is only 6%. Moreover, Figure 7b illustrates more than 92% and 95% of TP rate and TN rate, respectively for the CNN model. With 60%higher attack intensity, CNN false alerts are also less than 10%.

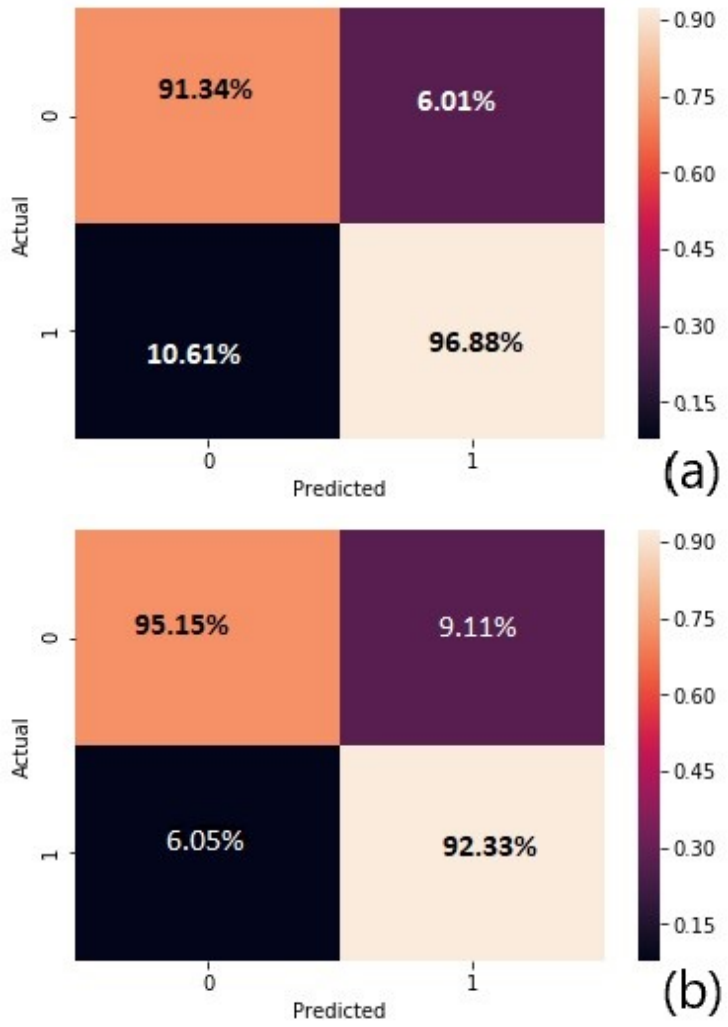

Figure 7. SAE and CNN Confusion matrix with 60% Attack Traffic. (**a**) Confusion matrix of the SAE Algorithm. (**b**) Confusion matrix of the CNN Algorithm.

Although SAE achieves an accuracy of 94%, on the other hand, CNN detection model also achieves nearly 93% detection accuracy. However, in this case, we combined more attack traffic as compared to benign traffic with 60% of malicious ratio, so both models perform well by utilising specific feature distributed traffic, which was obtained by entropy-based GE. In Table 7, we have depicted an average conclusion of a confusion matrix for SAE and CNN, addition to this, it also provided the accuracy results of both classifiers without utilising proposed entropy-based feature distribution for validation purpose.

We have validated our test-bed performance, with a comparison of selected classifier accuracy detection with GE entropy-based feature distribution and without entropy-based feature distribution, as illustrated in Figure 8. In Figure 8a, we provided SAE and CNN classifier accuracy results with 30% attack intensity. It can be observed that accuracy of both SAE and CNN classifiers with normal features is comparatively less than entropy-based features class. For 30% attack rate without entropy-based feature distribution, the CNN classifier achieves the average accuracy of around 62%, similarly, SAE classifier receives an average accuracy of 68%. However, when we performed classifiers test with GE entropy-based feature distribution with 30% of attack intensity, SAE classifier performed better with 84% accuracy scores, and CNN classifier was at an average of 77% accuracy score.

With 60% attack rate, both classifiers perform very well due to low false results, as depicted in Figure 12a, we observed total average rate for FP and FN as 6%, and 11%, respectively in SAE. However, we did not observe any drastic change in the results of CNN

evaluation, as CNN classifier also achieved FP alert with 9% and FN alert 7%. Overall, we observed that higher attack intensity enabled both classifiers to achieve significant results, but SAE achieved higher accuracy due to its lightweight processing.

Table 7. Performance metrics of confusion matrix for the SAE and CNN model.

SAE model (With feature distribution)	**30% attack traffic**	**60% attack traffic**
Avg. Accuracy	84%	94%
Avg. FP rate	8.9%	6.01%
Avg. FN rate	12.11%	10.6%
CNN model (With feature distribution)	**30% attack traffic**	**60% attack traffic**
Avg. Accuracy	77%	93%
Avg. FP rate	17%	9%
Avg. FN rate	10%	6%
SAE model (Without feature distribution)	**30% attack traffic**	**60% attack traffic**
Avg. Accuracy	68%	86%
CNN model (Without feature distribution)	**30% attack traffic**	**60% attack traffic**
Avg. Accuracy	62%	86%

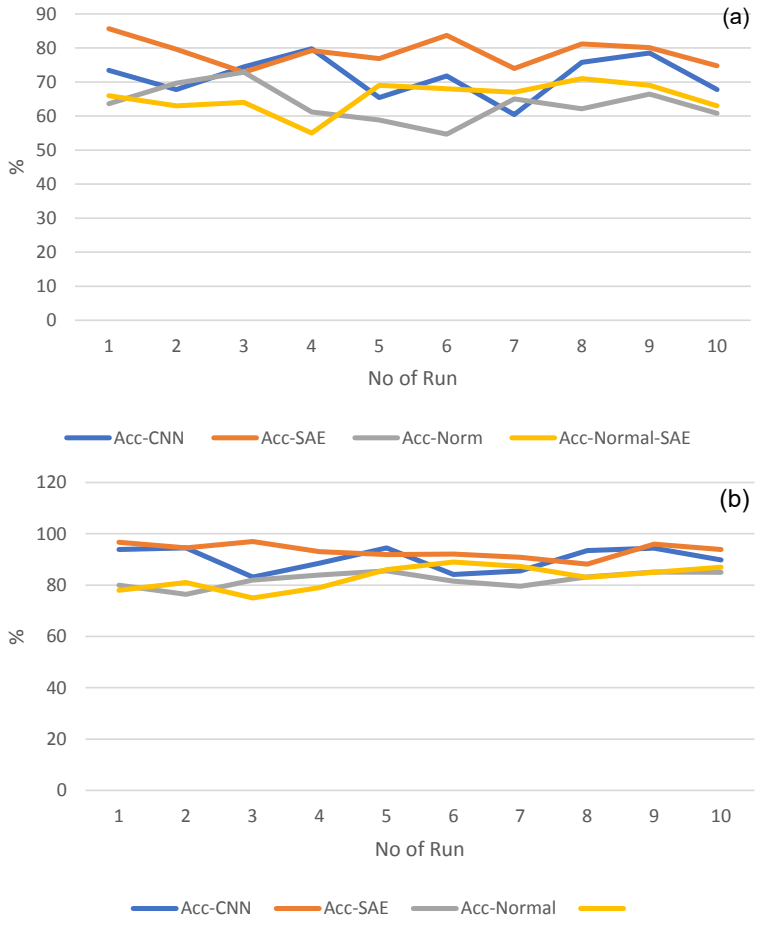

Figure 8. Accuracy comparison between Normal and Entropy based features. (**a**) With 30% attack rate accuracy comparison. (**b**) With 60% attack rate accuracy comparison.

Although, with the attack intensity of 60% as shown in Figure 8b, SAE and CNN accuracy was recorded as 94.3% and 93%, respectively, this result was obtained with entropy-based feature distribution. In contrast, we also run both classifiers before proposed entropy algorithm, the CNN classifier and SAE classifiers achieved an average

accuracy of 86%, which was nearly 10% accuracy less as compared to GE entropy-based feature distribution.

5.2. Performance Evaluation with CPU Usage at Controller

As our model is focused to improve Anti-DDoS model accuracy by reducing controller burden. We used combined entropy to distribute malicious features from data-plane to control-plane. In order to evaluate model performance we provided the comparison of CPU usage, after entropy calculation to distribute malicious traffic from test-bed, and before entropy calculation, which is depicted in Figures 9 and 10, respectively. During the event of DDoS attacks, control-plane was under heavy traffic as attack incoming-packets to SDN controller is measured in MB. Moreover, Shannon and Renyi entropy generalisation was deployed in the main controller, so that every attack incoming-packet was calculated with GE to distribute more specific features.

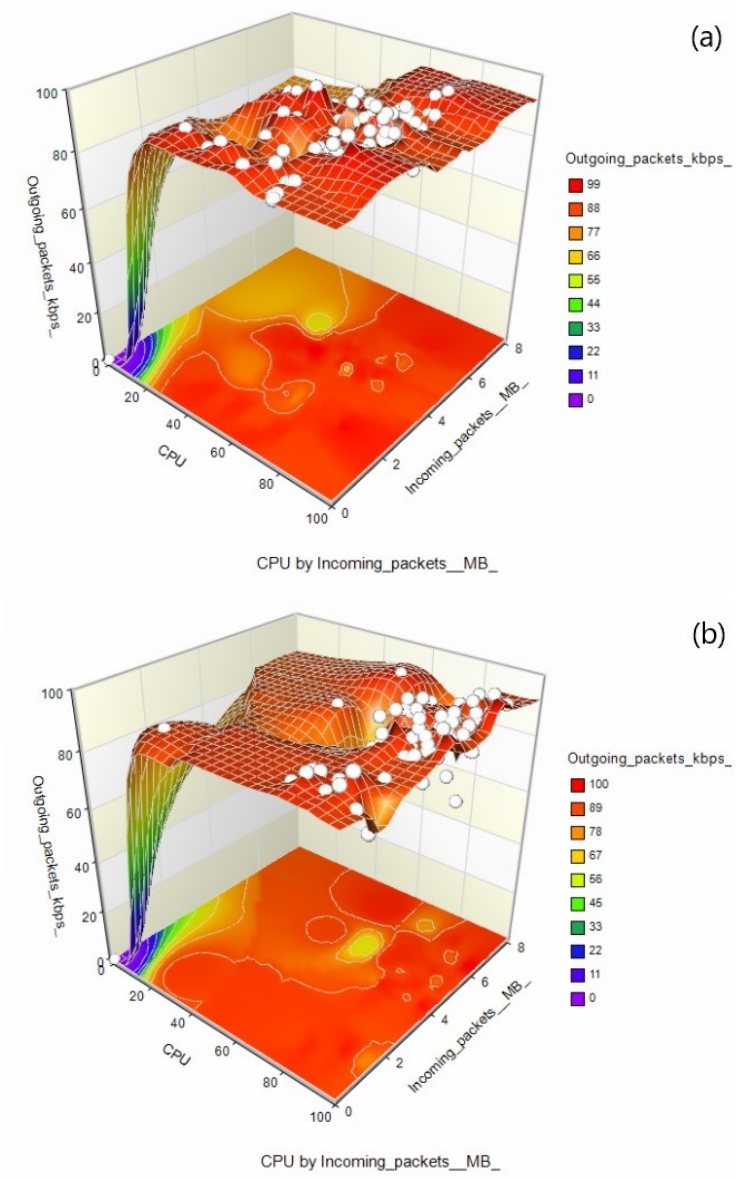

Figure 9. CPU utilisation after feature distribution with different attack rates. (**a**) 30% Attack Traffic CPU utilisation. (**b**) 60% Attack Traffic CPU utilisation.

The Figure 9a, represents the CPU utilisation with only 30% attack rate traffic with corresponding incoming-packets in megabits (mb) and outgoing-packets in kilobytes per second (kbps). In Figure 9a, the average CPU is utilised between 75% and 81% of the total.

As shown from the graph, when incoming-packets reached to an average of 4 Mbit/s, then nearly 80% of total CPU was used. When incoming-packets reduced to average of 2.6 Mbit/s after 5 s fraction then CPU fluctuated between 60% and 75%. At some time intervals of around every 55 s CPU utilization was as lower as 50%. Following entropy calculation, when we doubled the attack rate of 60% to our test-bed, then average CPU utilisation increased up to 85% and 93% as illustrated in Figure 9b. It represented around 93% of CPU usage during the event of around 9 Mbits/s incoming packets. When incoming-packets reduced to around 5 Mbit/s then CPU utilised was 85%. However, as the graph shows that we received only 60% of minimum CPU consumption. Outgoing-packets during both attack rates scenarios stood almost common, in Figure 9a,b, outgoing packets were changing between 80 and 96 kbps.

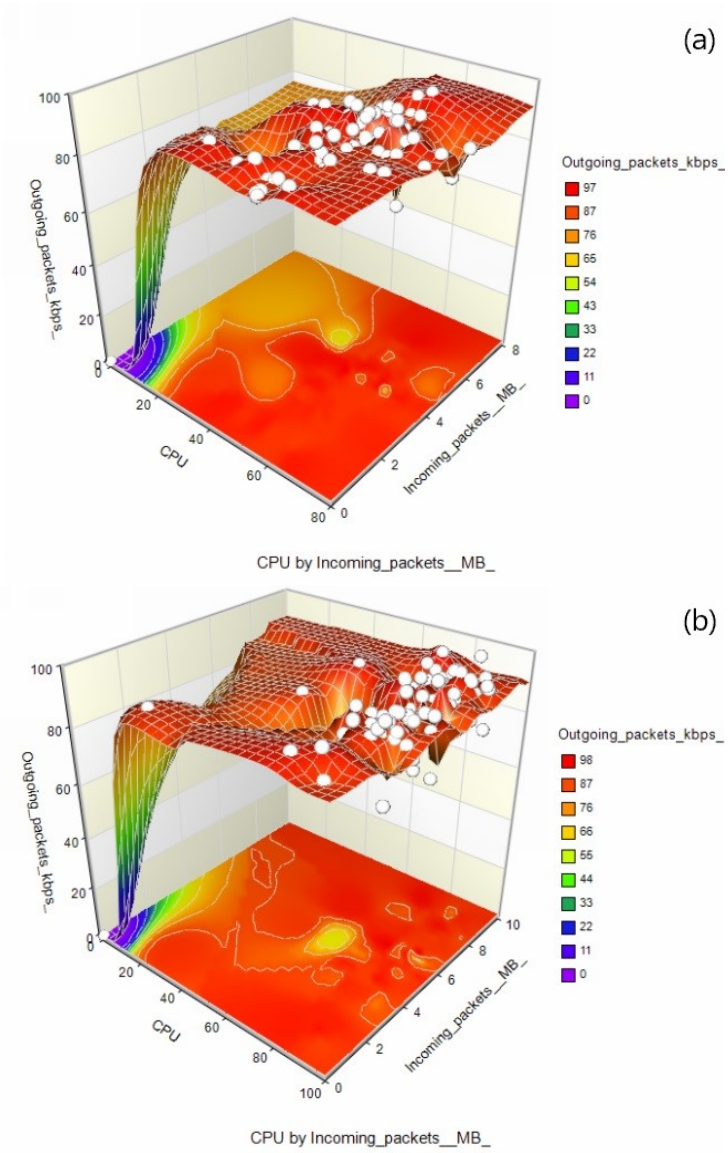

Figure 10. CPU utilisation without feature distribution with different attack rates. (**a**) 30%Attack Traffic CPU utilisation. (**b**) 60% Attack Traffic CPU utilisation.

In our work, we also calculated CPU usage before entropy-based feature distribution with respect to incoming packets and outgoing-packets of 30% and 60% attack rate traffic. The Figure 10a, shows CPU usage with only 30% attack rate without entropy calculation, the graph depicts the average CPU of 52% during the event of 3 Mbit/s incoming-packets towards proposed design. The CPU utilisation was 65% when incoming-packets reached

to around 5.5 Mbits/s level, minimum CPU usage was calculated 45% in some instances. Similarly, Figure 10b depicts CPU usage with 60% of attack rate without GE calculation. It shows nearly 80% of CPU was found busy with 5 Mbit/s then it frequently increased up to 90% of total CPU with 9.8 Mbit/s incoming-packets flows.

It can be seen from Figure 9a, feature distribution with entropy generalisation used 20% more CPU than the normal traffic, when we used low-intensity attack rate such as 30% attack rate. On the other hand, when we increased attack intensity to 60%, then average CPU was around 7%, after comparing values from Figures 9b and 10b. Overall, the distribution of features with entropy generalisation was not consuming more CPU resources with higher attack intensity in test-bed.

In another experiment, we evaluated results of FP and FN reports with 30% and 60% of attack rates, we provided a comparison between SAE and CNN classifiers. We run both algorithms 10 times to acquire different results as depicted in Figures 11 and 12. With 30% of attack rates, the average number of FP alerts and FN alerts were 10% and 12%, respectively for SAE classifier as illustrated in Figure 11a. Similarly, from Figure 11b, we can observe that 30% of attack rate in the classifier of CNN, represented the average rate of FP alerts as 17% and FN alerts as nearly 10%, which was slightly higher than SAE classifier.

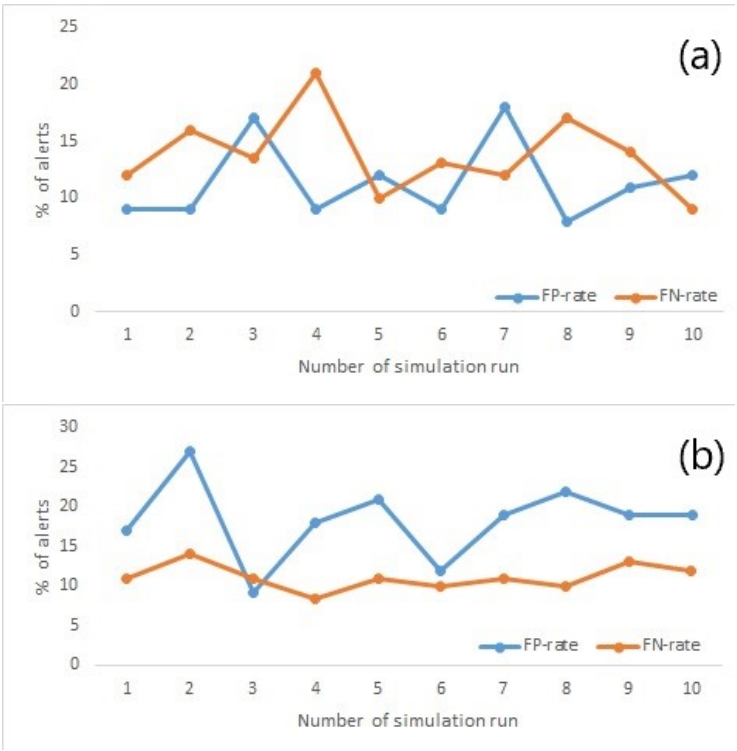

Figure 11. SAE and CNN False reports with 30% Attack Traffic. (**a**) SAE Algorithm False reports. (**b**) CNN Algorithm False reports.

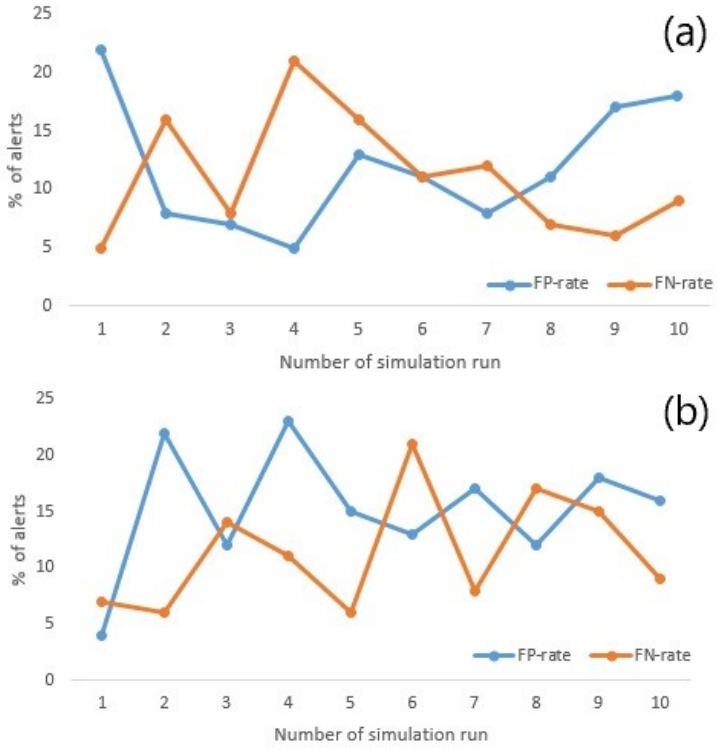

Figure 12. SAE and CNN False reports with 60% Attack Traffic. (**a**) SAE Algorithm False reports. (**b**) CNN Algorithm False reports.

6. Conclusions and Future Directions

In SDN-based environment, control-plane is always under severe threats, due to heavy flows from the attacker. Control-plane, centrally manages and manipulates packet-in handling, data collection, classification algorithms and other traffic manipulation tools. In the event of DDoS attack, it fails to run all these implemented elements which cause low accuracy and higher false alerts. For the anti-DDoS model, it is very important to identify attacks as soon as possible otherwise massive flows of packet-in events will start to deplete the controller resources. To overcome this issue, we have utilised feature distribution technique by entropy generalisation of Shannon and Renyi combined formula. Although Shannon entropy is already used in existence work for classification results, our work used Generalised Entropy (GE) for the purpose of Information Distance (ID). This combined entropy technique helps to reduce controller overhead as GE removes redundant and unnecessary traffic feature, which enables the SDN controller to identify attack packets effectively so that networks regardless of it size, can effectively mitigated DDoS attack. Entropy calculation utilises around 25% more CPU, when we merge 30% of attack traffic. Our implemented GE technique to distribute traffic features uses only 5% more CPU when attack intensity was 60% in test-bed. This is due to the fact that we fixed TW = 3 threshold, which is higher time windows and is feasible to manipulate and distribute huge attack traffic. Our work uses two well-known classifiers SAE, and CNN to perform classification with distributed traffic as provided in Table 3. With 60% attack intensity, SAE and CNN achieved an average accuracy of 94% and 93%, respectively with only 6% of FP alerts in SAE traffic classification.

In this work, the deployment of the NIDS was fixed. As future work, we will explore the impact of NIDS placement within the network on detection rate and accuracy. We will also explore the practical implications of using various types of network sensors used by the industry—open-source and proprietary.

Author Contributions: Conceptualization, R.M.A.U. and Z.P.; methodology, R.M.A.U. and K.D.; software, R.M.A.U.; validation, R.M.A.U., W.A.K. and A.M.K.; formal analysis, Z.P. and K.D.; investigation, R.M.A.U.; resources, Z.P.; data curation, R.M.A.U. and Z.P.; writing–original draft preparation, R.M.A.U. and W.A.K.; writing–review and editing, Z.P., A.M.K. and B.H.; visualization, R.M.A.U.; supervision, Z.P. and W.A.K.; project administration, K.D.; funding acquisition, W.A.K. and A.M.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Zayed University Cluster Research Fund grant number R#18038. and The APC was funded by R#18038.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sachdeva, M.; Kumar, K.; Singh, G.; Singh, K. Performance analysis of web service under DDoS attacks. In Proceedings of the 2009 IEEE International Advance Computing Conference, Patiala, India, 6–7 March 2009; pp. 1002–1007.
2. Alsmadi, I.; Xu, D. Security of software defined networks: A survey. *Comput. Secur.* **2015**, *53*, 79–108. [CrossRef]
3. Dargahi, T.; Caponi, A.; Ambrosin, M.; Bianchi, G.; Conti, M. A survey on the security of stateful sdn data planes. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1701–1725. [CrossRef]
4. Hu, F.; Hao, Q.; Bao, K. A survey on software-defined network and openflow: From concept to implementation. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 2181–2206. [CrossRef]
5. Sahoo, K.S.; Deepak, P.; Mohammad, S.; Obaidat, A.S.; Sambit, K.M.; Bibhudatta, S. On the placement of controllers in software-defined-WAN using meta-heuristic approach. *J. Syst. Softw.* **2018**, *145*, 180–194. [CrossRef]
6. Farhady, H.; Lee, H.; Nakao, A. Software-defined networking: A survey. *Comput. Netw.* **2015**, *81*, 9–95. [CrossRef]
7. Akhunzada, A.; Ahmed, E.; Gani, A.; Khan, M.K.; Imran, M.; Guizani, S. Securing software defined networks: Taxonomy, requirements, and open issues. *IEEE Commun. Mag.* **2015**, *53*, 6–44. [CrossRef]
8. Vissers, T.; Somasundaram, T.S.; Pieters, L.; Govindarajan, K.; Hellinckx, P. DDoS defense system for web services in a cloud environment. *Future Gener. Comput. Syst.* **2014**, *37*, 37–45. [CrossRef]
9. Mirkovic, J.; Reiher, P. A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Comput. Commun. Rev.* **2004**, *34*, 39–53. [CrossRef]
10. Wang, R.; Zhiping, J.; Lei, J. An entropy-based distributed DDoS detection mechanism in software-defined networking. In Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA, Washington, DC, USA, 20–22 August 2015; Volume 1, pp. 310–317.
11. Mehdi, S.A.; Khalid, J.; Khayam, S.A. Revisiting traffic anomaly detection using software defined networking. In *International Workshop on Recent Advances in Intrusion Detection*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 161–180.
12. Braga, R.; Mota, E.; Passito, A. Lightweight DDoS flooding attack detection using NOX/OpenFlow. In Proceedings of the IEEE Local Computer Network Conference, Denver, CO, USA, 10–14 October 2010; pp. 408–415.
13. Giotis, K.; Argyropoulos, C.; Androulidakis, G.; Kalogeras, D.; Maglaris, V. Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments. *Comput. Netw.* **2014**, *62*, 122–136. [CrossRef]
14. Shin, S.; Porras, P.A.; Yegneswaran, V.; Fong, M.W.; Gu, G.; Tyson, M. Fresco: Modular composable security services for software-defined networks. In Proceedings of the 20th Annual Network & Distributed System Security Symposium, San Diego, CA, USA, 24–27 February 2013.
15. Xiao, P.; Qu, W.; Qi, H.; Li, Z. Detecting DDoS attacks against data center with correlation analysis. *Comput. Commun.* **2015**, *67*, 66–74. [CrossRef]
16. Yu, M.; Jose, L.; Miao, R. Software Defined Traffic Measurement with OpenSketch. In Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13), Lombard, IL, USA, 2–5 April 2013; pp. 29–42.
17. Zhou, L.; Liao, M.; Yuan, C.; Zhang, H. Low-rate DDoS attack detection using expectation of packet size. *Secur. Commun. Netw.* **2017**, *2017*, 3691629:1–3691629:14 [CrossRef]
18. Kompella, R.R.; Singh, S.; Varghese, G. On scalable attack detection in the network. In Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement, Sicily, Italy, 25–27 October 2004; pp. 187–200.
19. Swain, B.R.; Sahoo, B. Mitigating DDoS attack and Saving Computational Time using a Probabilistic approach and HCF method. In Proceedings of the 2009 IEEE International Advance Computing Conference, Patiala, India, 6–7 March 2009; pp. 1170–1172.
20. Alanazi, S.; Al-Muhtadi, J.; Derhab, A.; Saleem, K.; AlRomi, A.N.; Alholaibah, H.S.; Rodrigues, J.J.P.C. On resilience of Wireless Mesh routing protocol against DoS attacks in IoT-based ambient assisted living applications. In Proceedings of the 2015 17th International Conference on E-health Networking, Application & Services (HealthCom), Boston, MA, USA, 14–17 October 2015; pp. 205–210.

21. Chen, Y.L.; Hwang, K. Collaborative detection and filtering of shrew DDoS attacks using spectral analysis. *J. Parallel Distrib. Comput.* **2006**, *66*, 1137–1151. [CrossRef]
22. Somani, G.; Gaur, M.S.; Sanghi, D.; Conti, M. DDoS attacks in cloud computing: Collateral damage to non-targets. *Comput. Netw.* **2016**, *109*, 157–171. [CrossRef]
23. Wang, H.; Jin, C.; Shin, K.G. Defense against spoofed IP traffic using hop-count filtering. *IEEE/ACM Trans. Netw.* **2007**, *15*, 40–53. [CrossRef]
24. Yan, Q.; Yu, F.R.; Gong, Q.; Li, J. Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges. *IEEE Commun. Surv. Tutor.* **2015**, *18*, 602–622. [CrossRef]
25. Li, J.; Cheng, K.; Wang, S.; Morstatter, F.; Trevino, R.P.; Tang, J.; Liu, H. Feature selection: A data perspective. *ACM Comput. Surv. (CSUR)* **2017**, *50*, 1–45. [CrossRef]
26. Romero, E.; Sopena, J.M. Performing feature selection with multilayer perceptrons. *IEEE Trans. Neural Netw.* **2008**, *19*, 431–441. [CrossRef]
27. Bolón-Canedo, V.; Sánchez-Maroño, N.; Alonso-Betanzos, A. A review of feature selection methods on synthetic data. *Knowl. Inf. Syst.* **2013**, *34*, 483–519. [CrossRef]
28. Kotani, D.; Okabe, Y. A packet-in message filtering mechanism for protection of control plane in OpenFlow switches. *IEICE Trans. Inf. Syst.* **2016**, *99*, 695–707. [CrossRef]
29. Dong, P.; Du, X.; Zhang, H.; Xu, T. A detection method for a novel DDoS attack against SDN controllers by vast new low-traffic flows. In Proceedings of the 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 23–27 May 2016; pp. 1–6.
30. Prokhorenko, V.; Choo, K.-K.R.; Ashman, H. Web application protection techniques: A taxonomy. *J. Netw. Comput. Appl.* **2016**, *60*, 95–112. [CrossRef]
31. Mohammadi, R.; Reza, J.; Mauro, C. Slicots: An sdn-based lightweight countermeasure for tcp syn flooding attacks. *IEEE Trans. Netw. Manag.* **2017**, *14*, 487–497. [CrossRef]
32. Osanaiye, O.; Cai, H.; Raymond Choo, K.; Ali, D.; Zheng, X.; Mqhele, D. Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing. *EURASIP J. Wirel. Commun. Netw.* **2016**. [CrossRef]
33. Ma, X.; Chen, Y. DDoS detection method based on chaos analysis of network traffic entropy. *IEEE Commun. Lett.* **2013**, *18*, 114–117. [CrossRef]
34. François, J.; Issam, A.; Raouf, B. FireCol: A collaborative protection network for the detection of flooding DDoS attacks. *IEEE/ACM Trans. Netw.* **2012**, *20*, 1828–1841. [CrossRef]
35. Yuan, T.; Yu, S. DDoS attack detection at local area networks using information theoretical metrics. In Proceedings of the 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, Melbourne, Australia, 16–18 July 2013; pp. 233–240.
36. Özçelik, İ.; Richard, R.B. Deceiving entropy based DoS detection. *Comput. Secur.* **2015**, *48*, 234–245. [CrossRef]
37. Cambiaso, E.; Gianluca, P.; Giovanni, C.; Maurizio, A. Slow DoS attacks: Definition and categorisation. *Int. J. Trust Manag. Comput. Commun.* **2013**, *1*, 300–319. [CrossRef]
38. Kandoi, R.; Antikainen, M. Denial-of-service attacks in OpenFlow SDN networks. In Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, Canada, 11–15 May 2015; pp. 1322–1326.
39. Lim, S.; Yang, S.; Kim, Y.; Yang, S.; Kim, H. Controller scheduling for continued SDN operation under DDoS attacks. *Electron. Lett.* **2015**, *51*, 1259–1261. [CrossRef]
40. Mousavi, S.M.; St-Hilaire, M. Early detection of DDoS attacks against SDN controllers. In Proceedings of the 2015 International Conference on Computing, Networking and Communications (ICNC), Garden Grove, CA, USA, 16–19 February 2015; pp. 77–81.
41. Xu, Y.; Choi, J.; Dass, S.; Maiti, T. *Introduction: Bayesian Prediction and Adaptive Sampling Algorithms for Mobile Sensor Networks*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 1–9.
42. Niyaz, Q.; Sun, W.; Javaid, A.Y. A deep learning based DDoS detection system in software-defined networking (SDN). *arXiv* **2016**, arXiv:1611.07400.
43. Chen, Z.; Jiang, F.; Cheng, Y.; Gu, X.; Liu, W.; Peng, J. Xgboost classifier for ddos attack detection and analysis in sdn-based cloud. In Proceedings of the 2018 IEEE International Conference on Big Data and Smart Computing (BigComp), Shanghai, China, 15–17 January 2018; pp. 251–256. [CrossRef]
44. David, J.; Thomas, C. Ddos attack detection using fast entropy approach on flow- based network traffic. *Procedia Comput. Sci.* **2015**, *50*, 30–36. [CrossRef]
45. Sahoo, K.S.; Tiwary, M.; Sahoo, B. Detection of high rate ddos attack from flash events using information metrics in software defined networks. In Proceedings of the 2018 10th International Conference on Communication Systems & Networks (COMSNETS), Bangalore, India, 3–7 January 2018; pp. 421–424.
46. Nanda, S.; Zafari, F.; DeCusatis, C.; Wedaa, E.; Yang, B. Predicting network attack patterns in SDN using machine learning approach. In Proceedings of the 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Palo Alto, CA, USA, 7–10 November 2016; pp. 167–172.

47. Kotani, D.; Okabe, Y. A packet-in message filtering mechanism for protection of control plane in openflow networks. In Proceedings of the 2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), Marina del Rey, CA, USA, 20–21 October 2014; pp. 29–40.
48. Liu, Y.; Ji, L.; Zhang, J.; Gao, C.; Qu, J. A Sensitivity Analysis of Attention-Gated Convolutional Neural Networks for Sentence Classification. *arXiv* **2019**, arXiv:1908.06263.
49. Wu, H.; Wang, C.; Tyshetskiy, Y.; Docherty, A.; Lu, K.; Zhu, L. The Vulnerabilities of Graph Convolutional Networks: Stronger Attacks and Defensive Techniques. *arXiv* **2019**, arXiv:1903.01610.
50. Lee, T.; Chang, L.; Syu, C. Deep Learning Enabled Intrusion Detection and Prevention System over SDN Networks. In Proceedings of the 2020 IEEE International Conference on Communications Workshops (ICC Workshops), Montreal, QC, Canada, 7–11 June 2020; pp. 1–6.
51. Rehman, R.U. *Intrusion Detection Systems with Snort: Advanced IDS Techniques Using Snort, Apache, MySQL, PHP, and ACID*; Prentice Hall Professional: Upper Saddle River, NJ, USA, 2003.
52. Zyczkowski, K. Rényi extrapolation of Shannon entropy. *Open Syst. Inform. Dynam.* **2003**, *10*, 297–310. [CrossRef]
53. Kumar, K.; Joshi, R.C.; Singh, K. A distributed approach using entropy to detect DDoS attacks in ISP domain. In Proceedings of the 2007 International Conference on Signal Processing, Communications and Networking, Chennai, India, 22–24 February 2007; pp. 331–337.
54. Behal, S.; Kumar, K. Detection of DDoS attacks and flash events using novel information theory metrics. *Comput. Netw.* **2017**, *116*, 96–110. [CrossRef]
55. Barron, A.R.; Gyorfi, L.; van der Meulen, E.C. Distribution estimation consistent in total variation and in two types of information divergence. *IEEE Trans. Inform. Theory* **1992**, *38*, 1437–1454. [CrossRef]
56. Kotsiantis, S.; Kanellopoulos, D.; Pintelas, P. Handling imbalanced datasets: A review. *GESTS Int. Trans. Comput. Sci. Eng.* **2006**, *30*, 25–36.