algorithms

MDPI

Article

An Adversarial DBN-LSTM Method for Detecting and Defending against DDoS Attacks in SDN Environments

Lei Chen [1,2], Zhihao Wang [3], Ru Huo [3,4,*] and Tao Huang [5]

[1] College of Compute, National University of Defense Technology, Changsha 410073, China
[2] Center for Strategic Studies, Chinese Academy of Engineering, Beijing 100088, China
[3] Purple Mountain Laboratories, Nanjing 211111, China
[4] Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China
[5] The State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
* Correspondence: huoru@bjut.edu.cn; Tel.: +86-182-1006-3771

Abstract: As an essential piece of infrastructure supporting cyberspace security technology verification, network weapons and equipment testing, attack defense confrontation drills, and network risk assessment, Cyber Range is exceptionally vulnerable to distributed denial of service (DDoS) attacks from three malicious parties. Moreover, some attackers try to fool the classification/prediction mechanism by crafting the input data to create adversarial attacks, which is hard to defend for ML-based Network Intrusion Detection Systems (NIDSs). This paper proposes an adversarial DBN-LSTM method for detecting and defending against DDoS attacks in SDN environments, which applies generative adversarial networks (GAN) as well as deep belief networks and long short-term memory (DBN-LSTM) to make the system less sensitive to adversarial attacks and faster feature extraction. We conducted the experiments using the public dataset CICDDoS 2019. The experimental results demonstrated that our method efficiently detected up-to-date common types of DDoS attacks compared to other approaches.

Keywords: software-defined networking; distributed denial of service; deep belief networks; long short-term memory

Citation: Chen, L.; Wang, Z.; Huo, R.; Huang, T. An Adversarial DBN-LSTM Method for Detecting and Defending against DDoS Attacks in SDN Environments. *Algorithms* **2023**, *16*, 197. https://doi.org/10.3390/a16040197

Academic Editor: Angelo Spognardi

Received: 1 February 2023
Revised: 27 March 2023
Accepted: 3 April 2023
Published: 5 April 2023

1. Introduction

Cyber Range [1] has become an essential piece of infrastructure as a significant scientific device for network security technology research, which is applied to network attack and defense drills, security product evaluation, network security personnel training, and network new technology verification. Cyber Range needs to construct NIDS to define DDoS attacks as the main target of DDoS attacks as an essential infrastructure.

At present, NIDSs have evolved to leverage ML models to cope with detecting new attacks. Some researchers have proposed highly accurate DDoS detection methods based on machine learning (ML) or deep learning (DL) for SDN. For instance, Hu et al. [2] proposed a method to realize UDP, ICMP, and IP spoofing attack detection by supporting the vector machine (SVM) classifier of ML. Abubakar and Pranggono [3] developed a flow-based intrusion detection system (IDS), which used neural network methods to detect attack flows. Niaz et al. [4] implemented a DDoS detection system using a stacked automatic encoder (SAE). Khamaiseh et al. [5] proposed a variational automatic encoder (VAE) classifier to detect saturation attacks for the OpenFlow switch. Although these schemes can achieve high detection accuracy for DDoS attacks, the above solutions are vulnerable to adversarial attacks [6–10]. Attackers inject small perturbed data streams to realize adversarial attacks, which aim to reduce the accuracy of malicious attack detection for the DL algorithm. Adversarial examples refer to samples with characteristic disturbances intended to cause

error classification of deep neural networks (DNN). The attackers can still successfully execute flooding DDoS attacks by adversarial attacks under IDS-based ML or DL.

The generator of the GAN model can generate many samples to construct an adversarial dataset. Using this dataset, we can avoid the challenges above in the DNN model. The GAN model is a deep generative model capable of learning the input data distribution, then producing new similar examples. A GAN consists of two models: a discriminator and a generator that are trained to compete with each other. The generator captures the training data distribution to produce fake samples resembling those from the training data. The discriminator needs to judge whether the received data are true or untruth. The generator and discriminator are constantly competing. Finally, the discriminator and a generator will reach a dynamic equilibrium. However, the added adversarial dataset expands the scale and dimension of data, and the DNN model may pay enormous computing costs to obtain suitable parameters. Obtaining or updating models is challenging and may cause overfitting, resulting in low scalability and a high error rate. We construct a DBN-LSTM model as the classifier model that can accurately detect DDoS attacks in the network and uses less time to train the model.

In this paper, we propose an adversarial DBN-LSTM method for detecting and defending against DDoS attacks in SDN environments, which can be divided into four steps: data collection, data processing, adversarial DBN-LSTM anomaly detection, and strategy distribution.

The main contributions of this paper are summarized as follows:

(1) We utilize the GAN model to generate many examples and construct an adversarial dataset. The perturbed samples from the adversarial dataset are added to the training data, which already contains normal and unperturbed DDoS samples, creating augmented data samples for adversarial training.
(2) We construct a DBN-LSTM model to represent the characteristics of DDoS attacks in SDN. This model extracts the time series information, completes the classification of dimension-reduction data, and significantly reduces the computational cost of large-scale data.

The rest of this paper is organized as follows: Section 2 describes the background of Detecting and Defending DDoS Attacks in SDN Environments. Section 3 illustrates the structure and principles of the proposed approach. The experimental results are shown in Section 4. Finally, the conclusion of our work is given in Section 5.

2. Related Work

Madry et al. [11] proposed the adversarial robustness of neural networks through the lens of robust optimization. They use a natural saddle point min–max formulation to capture the notion of security against adversarial attacks in a principled manner. Their work showed that the specially designed deep learning structure could effectively resist adversarial attacks.

Ujjan et al. [12] proposed sFlow and adaptive polling-based sampling with a Snort IDS and deep-learning-based model. Based on the flexible decoupling characteristics of SDN, this method can effectively reduce the impact of DDoS attacks. On the one hand, they deployed sFlow and adaptive polling-based sampling individually to reduce the calculation overhead of the SDN switch in the data plane. On the other hand, they optimized the SAE model and combined it with IDS to detect DDoS attacks in the control plane. Their work provides an excellent example of how to use the advantages of SDN to combine the deep learning model to realize the identification of DDoS attacks.

Zainudin et al. [13] proposed a low-cost approach for DDoS attack classification. This study combined CNN and LSTM and designed extreme gradient boosting. Their method can make the device not need to pay high computational power and can be well applied to IoT devices with low computational power. Their performance results show that the proposed model achieves high accuracy with a low time cost.

Razib et al. [14] proposed that the DL-model-driven SDN enabled IDS to be confronted with significantly less frequent attacks in SDN. In this study, the DNN model processes the data, and the corresponding data are sent into the designed LSTM model. The proposed model is trained on the CICIDS 2018 dataset. This model realized the detection of network attacks with higher accuracy and precision. However, this method does not take advantage of the characteristics of SDN and focuses more on improving the deep learning method itself.

Novaes et al. [15] proposed an IDS based on adversarial training in SDN. The difference from other studies is that this study further considers the impact of adversarial attacks on the system and implements the design of the GAN framework in the SDN environment. Their model avoids the impact of adversarial attacks and enables the model to detect DDoS attacks more accurately. Their experiments show higher accuracy compared with similar schemes with the public dataset CICDDoS 2019.

Alghazzawi et al. [16] proposed an effective hybrid DL model (CNN + BiLSTM) enhanced with the feature selection approach. This method built an x^2 test for feature selection and then utilized a CNN + BiLSTM hybrid model for DDoS attack classification. This method used the x^2 test to identify highly rated features that contribute considerably to predicting court case judgments and used a CNN to extract such high-rated features. Moreover, these features are then input into a BiLSTM model, which maintains the prior and future context of the provided data. This method can predict DDoS attack outcomes from data using both optimum feature selection and CNN and BILSTM layers.

Assis et al. [17] proposed an SDN defense system based on the analysis of single IP flow records, which uses the gated recurrent units (GRU) deep learning method to detect DDoS and intrusion attacks. This direct flow inspection enables faster mitigation responses, minimizing the attack's impact on the SDN. The proposed model is tested against several different machine learning approaches over two public datasets, the CICDDoS 2019 and the CICIDS 2018.

Javeed [18] developed an SDN-enabled DL-driven solution for the detection of threats in the IoT environment that is cost-effective and highly scalable. They used a hybrid technique comprising the Cuda-deep neural network long short-term memory (CuDNNLSTM) and Cuda-deep neural network gated recurrent unit (CuDNNGRU) algorithms for efficient threat detection. They implemented two algorithms for the comparison of our results: Cuda-deep neural network gated recurrent unit (CuDNN-GRU) and Cuda-bidirectional long short-term memory (CuBLSTM). Upon their comparison, they found their algorithm to perform both efficiently and accurately.

Saini [19] used a machine-learning-based approach to detect and classify different types of network traffic flows. The proposed approach is validated using a new dataset experiencing various modern types of attacks such as HTTP flood, SID DoS and normal traffic. They claimed that their algorithm produced the best results as compared to Random Forest and Naïve Bayes algorithms.

As mentioned, DL algorithms have shown enhanced results for detecting normal DDoS attacks. Furthermore, researchers improve DL algorithms or combine them with relevant technologies so that their solution can perform better in an SDN environment. However, these solutions encounter the problem of further increases in training data. That said, the authors of [20] have already impacted the problem of adversarial attacks, simply by applying the most basic GAN framework instead of improving it to make it applicable to the SDN environment. Moreover, Mittal et al. [21] proposed that the deep learning model's accuracy depends on the preprocessed data quality. Therefore, efficient training of the DL model requires suitable preprocessing techniques.

To solve these problems, we proposed an adversarial DBN-LSTM method for detecting and defending DDoS attacks in SDN environments.

3. Proposed System

We proposed architecture using an adversarial DBN-LSTM framework to detect and defend against DDoS attacks to protect the controllers.

As shown in Figure 1, we proposed a system that consisted of four independent modules in the application layer to protect the SDN controllers in the control plane (the source codes cannot be shared due to privacy and the development of our application). The system can be summarized as follows:

1. Data Collection: the SDN controller collects data from the physical and virtual switches in the data layer.
2. Data Processing: non-numerical features collected from the switch will be converted into numerical features.
3. Adversarial Deep Learning Anomaly Detection: this module will detect and mark the DDoS attack through the trained deep learning model.
4. Abnormal Defensing: this module will perform corresponding operations on DDoS attacks to avoid the harm of DDoS attacks.

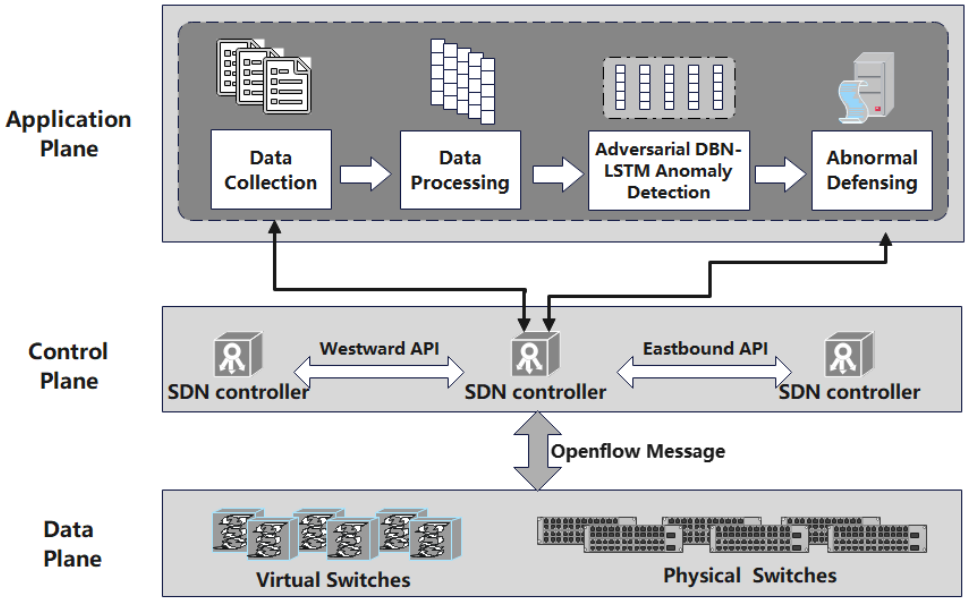

Figure 1. Proposed system architecture.

3.1. Data Collection

In this module, the SDN controller in the control layer will collect data from the virtual and physical switches in the data layer. This work provides preconditions for the subsequent detection of abnormal data. In the SDN environment, we use flows to define the data passing through the switch. The flows are a packet sequence passing an observation point in the network during a specific time interval. Previous works proposed that the data flow collection interval should generally be 1–5 min [17]. With the increase in network transmission bandwidth, it is feasible to reduce this time interval [18,19]. This paper defines the collection interval as 1 s to respond to the anomalies hidden in the rapid data flows.

After an interval t, the SDN controller will send requests to switches to collect IP flow records by OpenFlow messages. After the switches receive a request message from the SDN controller, they will send IP flow records in time intervals t. Figure 2 represents this process. This paper defines these IP flow records as a set $\beta_t = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, where each α_i is a flow record.

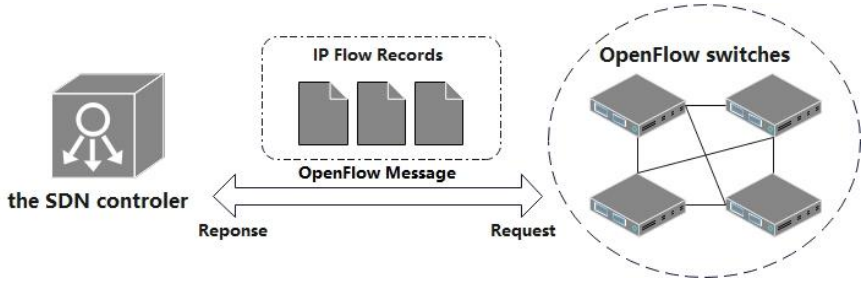

Figure 2. The process of data collection.

3.2. Data Processing

We will process non-numerical and numerical features from IP flow records in data preprocessing. For the non-numerical features, it is necessary to use one-hot encoding to transform the numerical features. For original numerical features, they need to be normalized in the range of (0,1) to reduce the impact of the different scales of features. The processing above assists the next module in the anomaly detection approach.

According to the public dataset CICDDoS 2019, this paper received 88 attributes from IP flow records as its features. The following attributes are included: bits/s, packets/s, Source IP Entropy, Source Port Entropy, Destination IP Entropy, Destination Port Entropy, and 82 other digital features, which include time stamp.

We define the number of bits and packets, source IP address and port number, and destination IP address and port number, which belong to α_i, as $k^{bits}_{\in\alpha_i}$, $k^{packets}_{\in\alpha_i}$, $k^{srcIP}_{\in\alpha_i}$, $k^{srcPort}_{\in\alpha_i}$, $k^{dstIP}_{\in\alpha_i}$, and $k^{dstPort}_{\in\alpha_i}$. Other 82 digital features are the same as these examples.

As non-numerical features, IP addresses and port numbers differentiate addresses. Under a DDoS attack occurrence, the destination IP address and destination Port number distribution may become concentrated due to the high number of connections requested by the attackers [15]. So, we need to process the data through the Shannon entropy formula to evaluate whether the address of IP flow records is centralized. We define $k^{srcIP}_{\in\alpha_i} = \{k_1, k_2, \ldots, k_n\}$ as a set of the source IP feature belonging to α_i. k_j means the frequency of occurrences of a source IP in α_i. S is the sum of all occurrences of all source IPs in α_i. n means the number of source IPs in α_i.

For each α_i, the Shannon entropy H(·) for the flow feature $k^{srcIP}_{\in\alpha_i}$ is defined as source IP entropy, which can be processed as follows:

$$H\left(k^{srcIP}_{\in\alpha_i}\right) = -\sum_{j=1}^{n}\left(\frac{k_j}{S}\right) log_2\left(\frac{k_j}{S}\right),$$

$$S = \sum_{j=1}^{n} k_j$$

The source port number, destination IP address, and port number are the same as the source IP.

As the numerical features, the rate of bits and packets per second can be processed as follows:

$$\sum_{j=1}^{n} k^{bits}_{\in\alpha_i};$$

$$\sum_{j=1}^{n} k^{packets}_{\in\alpha_i}.$$

Moreover, the other 82 digital features that can be processed are the same as the above features.

3.3. Adversarial DBN-LSTM Anomaly Detection

In this module, we will judge whether the IP flow record is abnormal according to its features by sending it into the adversarial DBN-LSTM model. Firstly, we propose an effective method to produce many normal, DDoS, and adversarial DDoS samples by the GAN model. Then, the Classifier model will be trained by the above samples. DBN and LSTM construct the Classifier model. In the following, we briefly introduce the training process based on the GAN and Classifier models used in this work. The whole training process is shown in Figure 3.

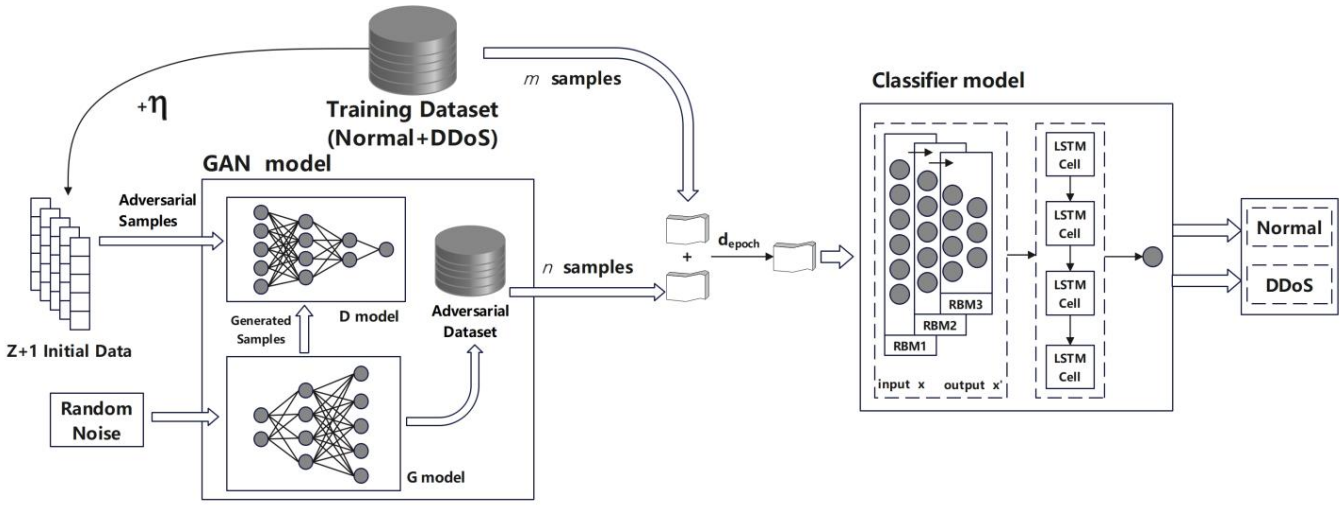

Figure 3. Adversarial DBN-LSTM training structure.

The GAN model mainly includes the generator (G) and discriminator (D) models. The G model needs to learn the original data distribution to generate untruth data to fake the D model. The discriminator needs to judge whether the received data are true or untrue.

The generated samples come from the G model by random noise in this paper. Furthermore, some of the data in the original dataset receive disturbance to generate adversarial samples. The generated samples and adversarial samples will be sent to the D model. Moreover, the D model determines the probability that the samples belong to the adversarial samples. This competition is equivalent to a minima x two-player game. The generator and discriminator are constantly competing. Finally, the G model and D model will reach a dynamic equilibrium.

The structure of the D model is the same as the G model. However, they have different objective functions and stochastic gradients.

For the D model, its objective function and stochastic gradients are presented as follows:

$$MaxE_{x\sim P_{data}}[logD(x)] + E_{y\sim P_G}[\log(1 - D(y))]$$

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

For the G model, its objective function and stochastic gradients are presented as follows:

$$min\ MaxE_{x\sim P_{data}}[logD(x)] + E_{y\sim P_G}[\log(1 - D(y))]$$

$$\nabla_{\theta_G}\frac{1}{m+1}\sum_{i=1}^{m+1}\left[\log\left(1 - D\left(y^i\right)\right)\right]$$

$x \sim P_{data}$ is the data from the original distribution, x^i is the No. i IP flow record, $D(x)$ is the probability of the original adversarial data, $y \sim P_G$ is the data from the generated distribution by G, and $D(y)$ is the probability of generated data. In particular, we obtained the initial $y \sim P_G$ based on the perturbation of $x \sim P_{data}$, which is based on the FGSM algorithm, as follows:

$$W^T \cdot y_1^i = W^T \cdot x^i + W^T \cdot \varepsilon sign(\Delta J_X(x, x^{\sim}, \theta))$$

The GAN model training steps are continuously implemented until the D model cannot identify the generated samples from the G model or reach t iterations training. The whole process is as Algorithm 1.

Algorithm 1 The GAN model training steps

Require: the G model; the D model; the Training dataset
1: Initialize θ_D and θ_G
2: **while** t iterations training or stop condition not met **do**
3: **for** d_{epoch} **do**
4: Sample $\{x^1, x^2, \ldots, x^{m+1}\}$ from P_{data}
5: Sample $\left\{y_1^1, y_1^2, \ldots, y_1^{m+1}\right\}$ from P_G
6: Generates $z^i = G\left(y^i\right)$ by G
7: Calculate $\theta_D = \theta_D - \eta \nabla_{\theta_D} \frac{1}{m+1} \sum_{i=1}^{m+1} \left[logD\left(x^i\right) + log\left(1 - D\left(z^i\right)\right)\right]$
8: **end for**
9: Sample $\left\{y_2^1, y_2^1, \ldots, y_2^{m+1}\right\}$ from P_G
10: Calculate $\theta_G = \theta_G - \eta \nabla_{\theta_G} \frac{1}{m+1} \sum_{i=1}^{m+1} \left[logD\left(x^i\right) + log\left(1 - D\left(y_2^i\right)\right)\right]$
11: **end while**
12:
13: return G and D

After reaching a dynamic equilibrium, the G model generates many samples, which are used to fool the Classifier model, to construct an adversarial dataset. Then, the Classifier model randomly extracts n samples from the adversarial dataset and m samples from the training dataset. After aggregating into a complete set, these samples will be extracted features by the Classifier model. We use the DBN algorithm to complete the process of extracting features. In this paper, we define $v^{(0)}$ as the visible layer. The complete set, which the Classifier model constructs, provides its data. For a neuron $h_j^{(0)}$ in the hidden layer, the activation function is as follows:

$$P\left(h_j^{(0)} \middle| v^{(0)}\right) = \sigma\left(W_j \cdot v^{(0)} + a_j\right)$$

In the above function, σ is the activation function, a_i is a bias coefficient and W_j is the weight. For these neurons in the same layer, their activation functions are calculated independently, so they can be identified as follows:

$$P\left(h^{(0)} \middle| v^{(0)}\right) = \Pi_{j=1}^{N} P\left(h_j^{(0)} \middle| v(0)\right)$$

In the above function, N is the number of neurons in the hidden layer. Then, we need to implement one-step Gibbs sampling to obtain the value. The calculation function is as follows:

$$h^{(0)} \sim P\left(h^{(0)} \middle| v^{(0)}\right)$$

Then, the visible layer and $h^{(0)}$ will reconstruct the following calculation, which is shown as follows:

$$P\left(v_i^{(1)} \middle| h^{(0)}\right) = \sigma\left(W_i^T \cdot v^{(0)} + b_i\right)$$

b_i is a bias coefficient. The activation functions are also calculated independently for the visible layer's probability distribution. So, they can be calculated as follows:

$$P\left(v^{(1)}\middle|h^{(0)}\right) = \Pi_{i=1}^{M} P\left(v_i^{(0)}\middle|h^{(0)}\right)$$

In the above formula, M is the number of neurons in the visible layer. We will implement one-step Gibbs sampling to obtain the value as well as the calculation function of $v^{(1)}$, as follows:

$$v^{(1)} \sim P\left(v^{(1)}\middle|h^{(0)}\right)$$

Then, we need to recalculate the activation functions of the hidden layer. The specific calculation is as follows:

$$P\left(h_i^{(1)}\middle|v^{(1)}\right) = \sigma\left(W_i^T \cdot v^{(1)} + a_i\right)$$

Finally, we can update the W, a, and b according to the previously calculated functions. The calculation is as follows:

$$W = W + \lambda \cdot \left(P\left(h^{(0)}\middle|v^{(0)}\right) \cdot v^{(0)T} - P\left(h^{(1)}\middle|v^{(1)}\right) \cdot v^{(1)T}\right)$$

$$a = a + \lambda \cdot \left(P\left(h^{(0)}\middle|v^{(0)}\right) - P\left(h^{(1)}\middle|v^{(1)}\right)\right)$$

$$b = b + \lambda \cdot \left(v^{(0)} - v^{(1)}\right)$$

In the above formula, λ is the learning rate of the DBN algorithm. When the parameters of W, a, and b are not changing, the parameters will stop updating. It means these features in the hidden layer can represent the data in the visible layer. Furthermore, the dimension of the hidden layer is significantly less than the visible layer, reducing the difficulty of data processing. After the DBN algorithm steps, the complete set aggregated by the Classifier model can effectively represent the original data with fewer dimension features.

Then, the extracted features and reduced dimensions will be sent to the LSTM. In the LSTM, in the training of the time sequence t, the output of the forget gate f_t is determined by the hidden state of the previous sequence h_{t-1} and current sequence data x_t as follows:

$$f_t = \sigma(W_t \cdot h_{t-1} + U_f \cdot x_t + b_f)$$

W_t and U_f represent the weights of the hidden state of the previous sequence and the current sequence data, respectively, and b_f is the bias coefficient of the forget gate. The output f_t represents the probability of forgetting the previous hidden cell state. Moreover, σ is the sigmoid function.

The input gate consists of two parts, one of which uses sigmoid as the activation function to obtain the intermediate variable i_t and the other of which uses tanh as the activation function to obtain the intermediate variable a_t, as follows:

$$i_t = \sigma(W_i \cdot h_{t-1} + U_i \cdot x_t + b_i)$$

$$a_t = \tan h(W_a \cdot h_{t-1} + U_a \cdot x_t + b_a)$$

The current cell state c_t will update by i_t, a_t and f_t as follows:

$$c_t = c_{t-1} * f_t + i_t * a_t$$

c_{t-1} is the previous hidden cell state. With the updated hidden cell state, the output of the LSTM unit can be calculated through the output gate, which is also the third gate in the LSTM unit. There are two outputs of the output gate as follows:

$$o_t = \sigma(W_o \cdot h_{t-1} + U_o \cdot x_t + b_o)$$

$$h_t = \tan h(W_h \cdot h_{t-1} + U_h \cdot x_t + b_h)$$

After obtaining the two variables, the output gate transmits them to the next time step (the next LSTM unit). In this paper, we use four-layer LSTM units to construct the LSTM network by training the accuracy of different numbers of layers of LSTM. After multiple training rounds, the Classifier model can output a judgment result about the IP flow records.

3.4. Abnormal Defending

Once the adversarial DBN-LSTM anomaly detection module detects a DDoS attack, this module will execute an exception prevention mechanism. Based on event-condition-action (ECA), we can set the corresponding trigger rules for the SDN controller at the application layer. Once the previous module detects a DDoS attack, it will send a message to the SDN controller to drop corresponding packets. We will set a list to record these IPs' ports to avoid some normal packets being discarded, similar to DDoS attacks. The mitigation process is summarized as follows in Algorithm 2.

Algorithm 2 Abnormal Defending Process

Require: Suspect flows β_t
1: Identify the suspect flows based on IP addresses and ports that make the analysis interval anomalous
2: Identify the destination IP address that receives the most flows
3: Identity in those flows the attackers' IP address, which has the same destination port
4: **If** IPs e ports are on the Safe List **then**
5: Forward packets
6: **Else**
7: Drop packets
8: End if

4. Experimental Results and Discussion

This section introduces evaluating indicators and experimental parameter settings. In addition, we selected some documents of the same type for performance comparison. Our experimental environment is a computer with an intel Core 12700H processor with fourteen cores, 16 GB RAM, and NVIDIA® GeForce GTX 3060 6 GB laptop GPU.

4.1. Evaluating Indicator

To better compare the performance with similar papers, we adopt four traditional indicators: *Accuracy* (A), *Precision* (P), *Recall* (R), and *F*1 Score (F1). The A represents the probability of the IP flow records being correctly identified. The P represents the percentage of actual DDoS attacks in determined DDoS attacks. The R represents the percentage of DDoS attacks detected in all DDoS attacks. As for F1, it is the harmonic between R and P. Their specific formula is defined as follows:

$$Accracy = \frac{TP + TN}{TP + FP + TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

TP means True Positive. *TN* represents True Negative. *FP* means False Positive. Moreover, *FN* means False Negative.

4.2. Experimental Setup

To improve the performance of the adversarial DBN-LSTM anomaly detection module, we evaluate the impact of the minibatch size, the learning rate on the DBN algorithm, and the number of layers of LSTM in the Classifier model. As for the GAN model, part parameters were set as dropout with 0.1, the learning rate with 0.001, and the optimizer with Adam. These parameters can achieve quite good performance.

Firstly, we set six different minibatch sizes to evaluate the length of epoch required for the convergence of different sizes. As shown in Figure 4, when we set the minibatch size to 32, we can obtain convergence with 86 batches. The convergence speed is superior to other minibatch sizes.

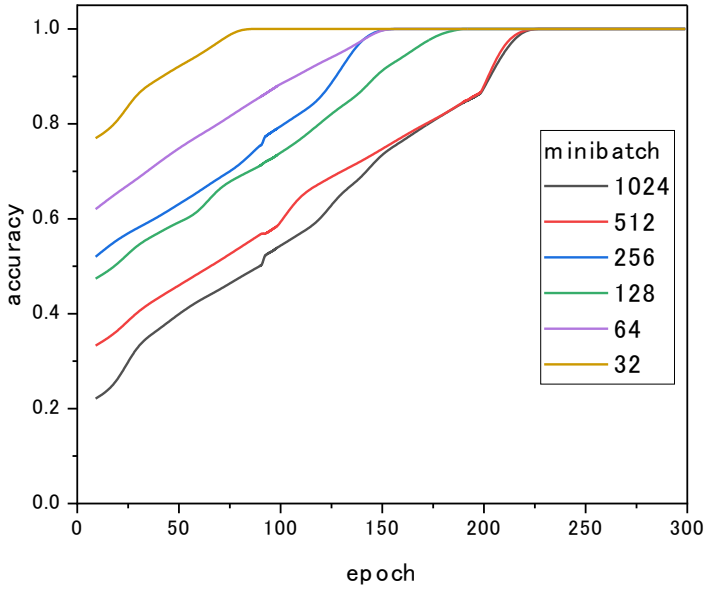

Figure 4. Accuracy of different Minibatch sizes.

Then, we compare the effect of the learning rate on the convergence training time. As shown in Figure 5, we can obtain the best result when we set the learning rate as 1.00. Therefore, the learning rate should be set as 1.00.

Finally, we need to determine the effect of an LSTM structure with different layers on the results. We selected LSTM structures with two, three, and four layers for a performance test in the rate of decline of training loss and the training accuracy of discrimination. As shown in Figures 6 and 7, four layers of LSTM obtain the best performance.

To avoid the difference between the training and test datasets, we compare the accuracy of LSTM structures with different layers in the test dataset. As shown in Figure 8, four-layer LSTM obtained higher accuracy than other LSTM structures. Therefore, we use a four-layer LSTM to construct the Classifier model with DBN.

4.3. Experimental Results

To accurately evaluate the performance comparison of this paper in real situations, we select CICDDoS2019. The CICDDoS 2019 dataset separates the data into two days. The first one is a training day, containing 12 types of different DDoS attacks, including DNS, MSSQL, Syn, NetBIOS, LDAP, SNMP, NTP, UDP-Lag, SSDP, WebDDoS, TFTP and UDP. The second is a testing day, containing six different types of DDoS attacks, which are Syn, UDP, NetBIOS, LDAP, UDP-Lag, and MSSQL. To make it easier for readers to understand, the above description has been added to the paper. The dataset uses CICFlowMeter to extract more than 80 features from the dataset and has a total of 50063112 records, including

50006249 DDoS attacks and 56863 normal samples. As for the evaluating indicators, we have given these indicators in Section 4.1. They are the Accuracy, Precision, Recall, and F1 Score.

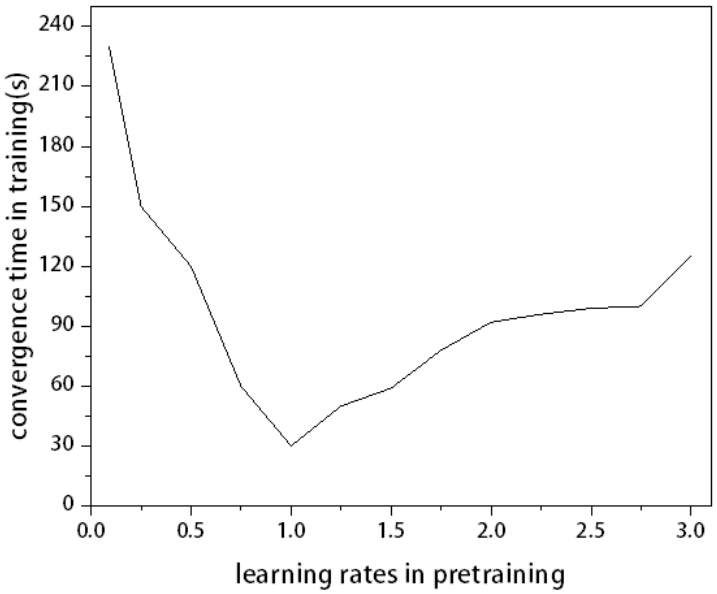

Figure 5. Convergence time in training with different learning rates in pretraining.

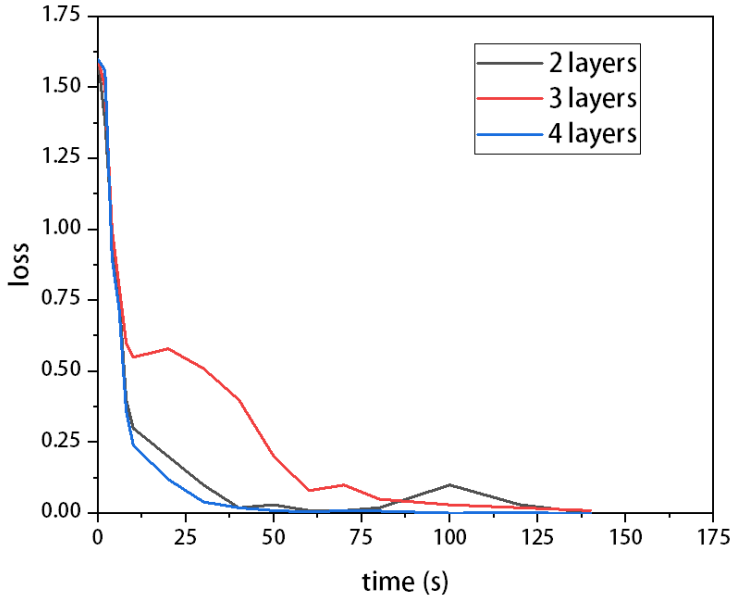

Figure 6. Loss with different numbers of layers of LSTM in the training dataset.

We select four representative deep learning methods from GAN [15], CNN [22], LSTM [23], and MLP [24]. For the above paper, we reset and verified the results based on CICDDoS2019. The division of the training set and the test set is the same as that of this article. Moreover, we have not changed the relevant parameter settings. Figure 9 shows that we give comparison results based on the evaluating indicators. Additionally, we took the average of five experiments after stabilization.

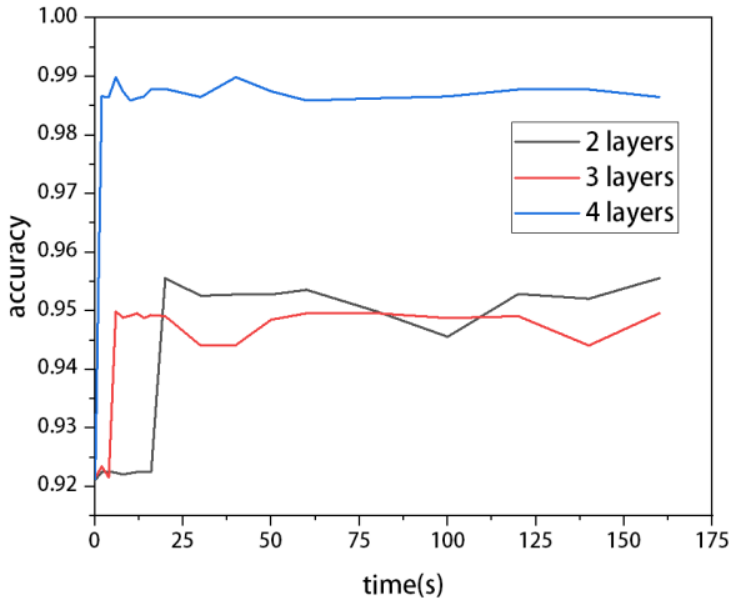

Figure 7. Accuracy of different numbers of layers of LSTM in the training dataset.

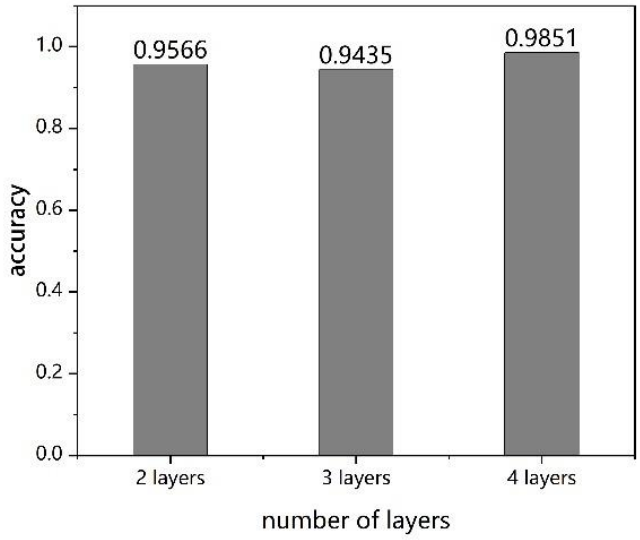

Figure 8. Accuracy of different numbers of layers of LSTM in the test dataset.

For Accuracy, our method reached an accuracy rate of 96.55%, which is higher than other methods. In the evaluating indicator, refs. [15,22] obtained virtually identical results (their difference is less than 0.12%). For [23,24], they reached 90.39% and 92.12%, respectively. For Precision, our method reached a rate of 96.44%. Ref. [22] reached a similar result (0.236% less than our method). Refs. [15,23,24] achieved results of 94.08%, 90.12%, and 92.12%, respectively. Like Recall, our method achieves a rate of 98.53% for the evaluating indicator. Compared with [15,22–24], our method is 0.64%, 6.12%, 9.1%, and 13.75% higher than these methods. As the harmonic between Recall and Precision, our method reaches the best result with 97.47%. The methods of [15,22–24] reach 95.94%, 92.81%, 89.77%, and 88.80%, respectively.

Furthermore, we construct a testing dataset of adversarial DDoS attacks based on the testing day of CICDDoS2019 using the FGSM algorithm. We used this dataset to test whether our method can resist adversarial attacks. We compared the DBN-LSTM and LSTM models without GAN. As shown in Table 1, the accuracy of our method in identifying

adversarial DDoS attacks can reach 91.23%. The DBN-LSTM and LSTM models without GAN can hardly resist adversarial DDoS attacks (less than 8%).

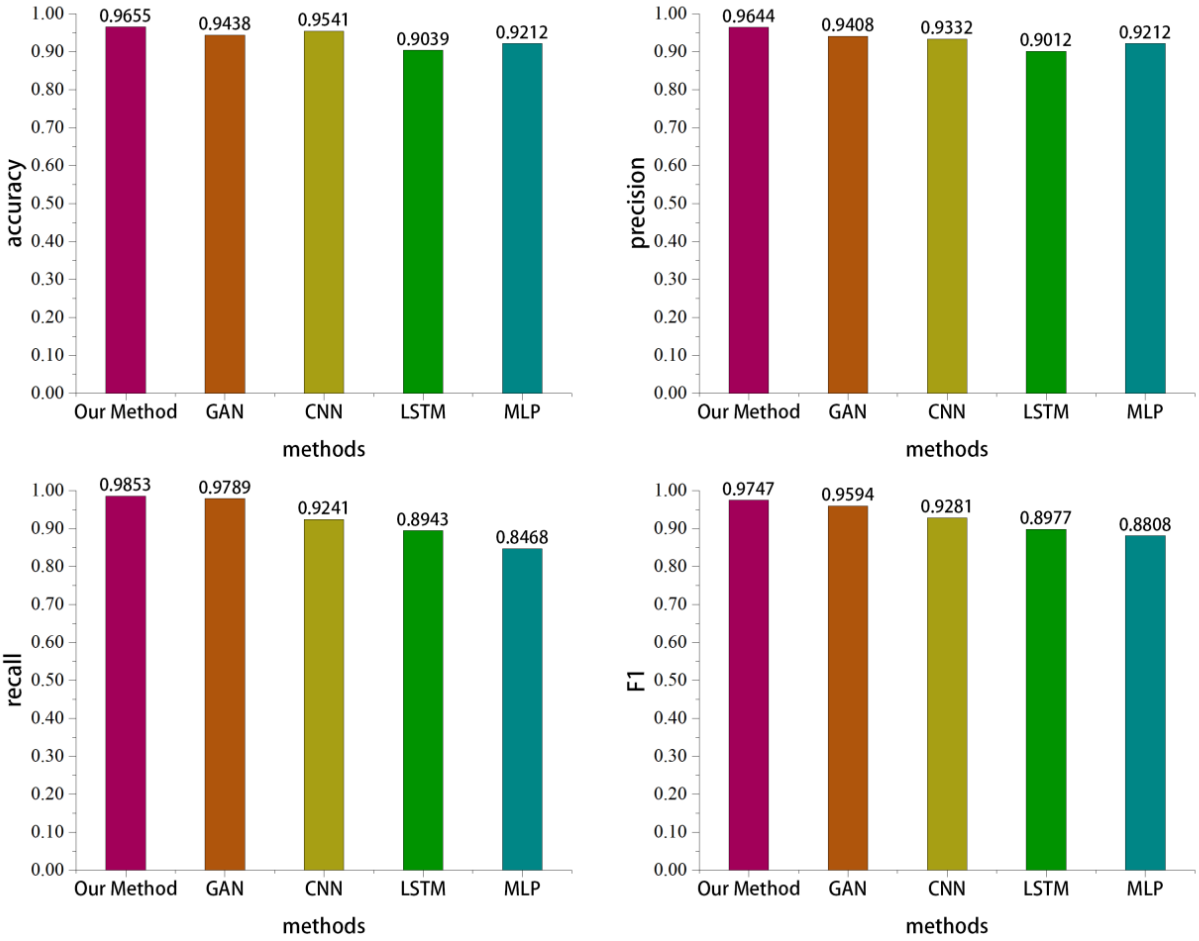

Figure 9. Comparison outcomes between our method and the compared methods through the evaluating indicator.

Table 1. Comparison outcomes on our method, DBN + LSTM, and LSTM.

	Accuracy	False
Our method	91.23%	8.77%
DBN + LSTM	7.62%	92.38%
LSTM	6.34%	93.77%

As mentioned above, we reached the best outcomes on the evaluating indicators. Our proposed method can effectively detect DDoS attacks and adversarial DDoS attacks.

5. Conclusions

This paper proposed an adversarial DBN-LSTM method for detecting and defending DDoS attacks. We designed four modules on the application plane: data collection, data processing, adversarial DBN-LSTM anomaly detection, and abnormal defending. First, the SDN controller realizes the fine-grained monitoring and collection of IP flow records in a complex network environment. Then, the IP flow records are preprocessed by normalization and the Shannon entropy formula. Finally, we designed the adversarial DBN-LSTM anomaly detection method. We can quickly generate many adversarial samples and build adversarial datasets using the GAN model. Furthermore, we introduced DBN for data dimensionality reduction and used LSTM to extract sample timing features to detect IP

flow records and identify adversarial DDoS attacks. Finally, we designed an abnormal defending strategy to provide SDN controllers. Our experiments show that our method can effectively detect DDoS attacks and make the system less sensitive to adversarial attacks.

Author Contributions: Writing—original draft preparation, L.C. and Z.W.; writing—review and editing, R.H.; supervision, T.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Key R&D Program of China, grant number 2019YFB1804403.

Data Availability Statement: The data are not publicly available due to privacy.

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

References

1. Mudassar, M.; Katt, B.; Gkioulos, V. Cyber ranges and security testbeds: Scenarios, functions, tools and architecture. *Comput. Secur.* **2020**, *88*, 101636.
2. Hu, D.; Hong, P.; Chen, Y. Fadm: Ddos flooding attack detection and mitigation system in software-defined networking. In Proceedings of the 2017 IEEE Global Communications Conference, Singapore, 4–8 December 2017.
3. Abubakar, A.; Pranggono, B. Machine learning based intrusion detection system for software defined networks. In Proceedings of the Seventh International Conference on Emerging Security Technologies (EST), Canterbury, UK, 6-8 September 2017.
4. Niyaz, Q.; Sun, W.; Javaid, A.Y. A deep learning based ddos detection system in software-defined networking (SDN). *arXiv* **2016**, arXiv:1611.07400. [CrossRef]
5. Khamaiseh, S.; Serra, E.; Xu, D. vswitchguard: Defending openflow switches against saturation attacks. In Proceedings of the IEEE Computer Society Signature Conference on Computers Software and Applications (COMPSAC), Madrid, Spain, 13–17 July 2020.
6. Shieh, C.S.; Nguyen, T.T.; Lin, W.W.; Lai, W.K.; Horng, M.F.; Miu, D. Detection of Adversarial DDoS Attacks Using Symmetric Defense Generative Adversarial Networks. *Electronics* **2022**, *11*, 1977. [CrossRef]
7. Jiang, H.; Lin, J.; Kang, H. FGMD: A robust detector against adversarial attacks in the IoT network. *Future Gener. Comput. Syst.* **2022**, *132*, 194–210. [CrossRef]
8. Nguyen, T.N. The challenges in ml-based security for SDN. In Proceedings of the 2018 2nd Cyber Security in Networking Conference (CSNet), Paris, France, 24-26 October 2018.
9. Papernot, N.; McDaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z.B.; Swami, A. The limitations of deep learning in adversarial settings. In Proceedings of the 2016 IEEE European symposium on security and privacy (EuroS&P), Saarbrucken, Germany, 21–24 March 2016.
10. Khamaiseh, S.Y.; Alsmadi, I.; Al-Alai, A. Deceiving Machine Learning-Based Saturation Attack Detection Systems in SDN. In Proceedings of the 2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Chandler, AZ, USA, 14–16 November 2020.
11. Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv* **2019**, arXiv:1706.06083.
12. Ujjan, R.M.A.; Pervez, Z.; Dahal, K.; Bashir, A.K.; Mumtaz, R.; González, J. Towards sFlow and adaptive polling sampling for deep learning based DDoS detection in SDN. *Future Gener. Comput. Syst.* **2021**, *125*, 156–167. [CrossRef]
13. Zainudin, A.; Ahakonye, L.A.C.; Akter, R.; Kim, D.-S.; Lee, J.-M. An Efficient Hybrid-DNN for DDoS Detection and Classification in Software-Defined IIoT Networks. *IEEE Internet Things J.* **2022**. [CrossRef]
14. Al Razib, M.; Javeed, D.; Khan, M.T.; Alkanhel, R.; Muthanna, M.S.A. Cyber Threats Detection in Smart Environments Using SDN-Enabled DNN-LSTM Hybrid Framework. *IEEE Access* **2022**, *10*, 53015–53026. [CrossRef]
15. Novaes, M.P.; Carvalho, L.F.; Lloret, J.; Proença, M.L., Jr. Adversarial Deep Learning approach detection and defense against DDoS attacks in SDN environments. *Future Gener. Comput. Syst.* **2021**, *125*, 156–167. [CrossRef]
16. Alghazzawi, D.; Bamasag, O.; Ullah, H.; Asghar, M.Z. Efficient detection of DDoS attacks using a hybrid deep learning model with improved feature selection. *Appl. Sci.* **2021**, *11*, 11634. [CrossRef]
17. Assis, M.V.; Carvalho, L.F.; Lloret, J.; Proença, M.L., Jr. A GRU deep learning system against attacks in software defined networks. *J. Netw. Comput. Appl.* **2021**, *177*, 102942. [CrossRef]
18. Javeed, D.; Gao, T.; Khan, M.T. SDN-enabled hybrid DL-driven framework for the detection of emerging cyber threats in IoT. *Electronics* **2021**, *10*, 918. [CrossRef]
19. Saini, P.S.; Behal, S.; Bhatia, S. Detection of DDoS attacks using machine learning algorithms. In Proceedings of the 2020 7th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 12–14 March 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 16–21.
20. Carvalho, L.F.; Abrão, T.; de Souza Mendes, L.; Proença, M.L., Jr. An ecosystem for anomaly detection and mitigation in software-defined networking. *Expert Syst. Appl.* **2018**, *104*, 121–133. [CrossRef]

21. Mittal, M.; Kumar, K.; Behal, S. Deep learning approaches for detecting DDoS attacks: A systematic review. *Soft Comput.* **2022**, 1–37. [CrossRef] [PubMed]
22. de Assis, M.V.; Carvalho, L.F.; Rodrigues, J.J.; Lloret, J.; Proença, M.L. Near realtime security system applied to SDN environments in IoT networks using convolutional neural network. *Comput. Electr. Eng.* **2020**, *86*, 106738. [CrossRef]
23. Priyadarshini, R.; Barik, R.K. A deep learning based intelligent framework to mitigate DDoS attack in fog environment. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 825–831. [CrossRef]
24. Wang, M.; Lu, Y.; Qin, J. A dynamic MLP-based DDoS attack detection method using feature selection and feedback. *Comput. Secur.* **2020**, *88*, 101645. [CrossRef]