Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 19(2): 1280–1303.
DOI: 10.3934/mbe.2022059
Received: 20 September 2021
Accepted: 24 November 2021
Published: 02 December 2021

Research article

IFACNN: efficient DDoS attack detection based on improved firefly algorithm to optimize convolutional neural networks

Jiushuang Wang, Ying Liu*and Huifen Feng

National Engineering Laboratory on Interconnection Technology for Next Generation Internet, Beijing Jiaotong University, Beijing, China

* **Correspondence:** Email: yliu@bjtu.edu.cn.

Abstract: Network security has become considerably essential because of the expansion of internet of things (IoT) devices. One of the greatest hazards of today's networks is distributed denial of service (DDoS) attacks, which could destroy critical network services. Recent numerous IoT devices are unsuspectingly attacked by DDoS. To securely manage IoT equipment, researchers have introduced software-defined networks (SDN). Therefore, we propose a DDoS attack detection scheme to secure the real-time in the software-defined the internet of things (SD-IoT) environment. In this article, we utilize improved firefly algorithm to optimize the convolutional neural network (CNN), to provide detection for DDoS attacks in our proposed SD-IoT framework. Our results demonstrate that our scheme can achieve higher than 99% DDoS behavior and benign traffic detection accuracy.

Keywords: software-defined internet of things; distributed denial of service; firefly algorithm; convolutional neural network; detect attacks

1. Introduction

Recently, network security has become particularly important because distributed denial of service (DDoS) [1] gravely threaten network safety. With the aim of dissipating the computing resources of the victim, for instance, memory and CPU, the DDoS attacker intentionally sends a good supply of malicious request messages to the victim host, making the affected host unable to provide services to normal users. DDoS attacks have increased dramatically in complexity, number, and impact due to the rapid growth of hire attack services and the internet of things (IoT) devices [2]. Service providers and network operators have done incalculable damage as DDoS attacks have become so devastating. In October 2016, Mirai Botnet ordered a large quantity of IoT equipment to launch a DDoS attack on the Dyn DNS infrastructure, leaving amazon, Twitter, GitHub and other prevalent Internet services unreachable for several hours [3]. The DDoS attack with a topmost traffic of 1.35 Tbps is believed to be the largest DDoS attack in history. The amount of IoT equipment with risk factors has increased dramatically, and there will be 24.6 billion connected devices by 2025 [4]. These IoT devices will improve the capabilities of extensive DDoS attacks. Currently, the incidents of DDoS attacks on IoT devices are increasing, and causing many IoT devices to fail and privacy leaks. Therefore, with the rapid development of the IoT, corresponding network security measures should also be improved simultaneously.

Software-defined networks (SDN) offers a novel opportunity to settle the above matter. In view of the success of SDN in network management and security maintenance, more and more domestic and foreign scholars have tried to introduce their design concepts into IoT and proposed a software-defined internet of things (SD-IoT) architecture [5]. The key characteristic of the SD-IoT architecture is the separation of the control plane and the forwarding plane. The SD-IoT controller Usually works on a high-performance computing platform, which enables the security strategies and detection mechanisms that cannot be achieved by traditional network [6]. For SD-IoT switches, the most important task is to deal with the traffic flow. The southbound interface brings the advantages of simplified equipment management and business implementation cost reduction to the system.This separation guarantees the continuous availability of IoT equipment and circumvents potential service failures and network outages. Moreover, it can prevent unauthorized admission to other devices from logging in, inspect devices that disrupt the Internet, distinguish normal and abnormal traffic flowing through IoT devices, and reduce the security risks of IoT devices eventually. In addition , more research is also essential because many problems such as network security still exist in the fusion of SDN and IoT.

DDoS attacks are currently one of the most difficult methods to detect in network attacks [7]. The purpose is to exhaust the target system or network resources, causing the victim to be unable to carry out normal services. Common DDoS attacks fall into two categories: resource bandwidth consuming attacks and system resource consuming attacks. Resource bandwidth attacks use a large number of zombie hosts to quickly generate a huge amount of traffic to converge on the victim's server, and completely seize its network bandwidth resources. For example, continuously sending a large number of UDP, TCP and ICMP packets can initiate a flooding attack, resulting in UDP flooding, TCP flooding and ICMP flooding. Or use reflection to perform amplification attacks, such as DNS reflection amplification attacks. System resource attacks mainly use protocol vulnerabilities to consume specific resources of the victim's host. For instance, TCP SYN half-connection attack using TCP three-way handshake.

At the same time, because of the many different combinations of DDoS attacks, detection becomes increasingly arduous. For instance, lots of DDoS attackers leverage mixed protocol packets to attack their victims. More comprehensive and persuasive defense techniques should be developed to deal with a variety of attack techniques [8]. The novel DDoS attacks cannot be detected by the traditional signature-based detection methods, while the more commonly used detection methods based on statistical anomalies are restricted by the detection threshold. In order to solve the deficiencies of statistical anomaly detection methods, attack detection schemes based on machine learning methods are being studied. Among them, deep learning algorithms have been recognized for the classification of DDoS attacks and normal traffic. The deep learning algorithms can extract the characteristics needed by DDoS attack and normal traffic flow from the original data flow. However, most of the attack detection methods based on deep learning algorithm in the past are implemented in traditional networks and require too much resource supply. Particularly, current DDoS attack detection methods are not designed for SD-IoT network offline attack detection. In live SD- IoT networks, detection algorithms must deal with networking traffic flows that have been predefined data packet window.

Although the large-scale CNN algorithm has achieved pretty results in detecting attacks, few people pay attention to how to maintain respectable detection performance with the supply of few resources. For example, deploy the trained CNN model in the SD-IoT controller. As more and more IoT equipment are deployed in the network, the probability of the network being attacked by insecure IoT equipment is also increasing, so we need to consider the defense mechanism. Deep learning approach can automatically extract high-level features from low-level ones and gain powerful representation and inference. We use deep learning for feature reduction of a large set of features derived from networking traffic. Thus, we choose the deep learning algorithm to apply to our scheme. In fact, even without resource constraints, it is valuable to minimize resource usage to maximize system output.

According to the latest DDoS detection requirements, this paper proposes a new detection algorithm, which integrates CNN algorithm into SD-IoT controller. The following highlight are the main contributions of this paper:

1) We put forward a SD-IoT security framework. This architecture includes IoT infrastructure, IoT switches that fuse IoT gateways, and an SD-IoT controller.

2) A dataset-independent data packet preprocessing mechanism in which detection algorithms must deal with flows fragments collected in predefined packet windows. The SD-IoT controller that provides flexible programmability periodically obtains the packet header from the SD-IoT switch, which greatly reduces the processing overhead of the SD-IoT controller.

3) We propose an improved firefly algorithm to optimize the neural network structure to improve the detection accuracy.

4) The detection method in this paper uses CNN algorithm to learn normal traffic and malicious traffic, and then detects DDoS attack. The method has high detection accuracy and low processing overhead.

Roadmap: In Section 2, we retrospect the related work. Section 3 presents the currently generally recognized IoT architecture, interprets the proposed SD-IoT framework and introduces about the DDoS attacks. In Section 4, we propose to detect DDoS attacks in SD-IoT environment with IFACNN algorithm. The simulation platform is used for experiments and performance assessment in Section 5. Lastly, Section 6 gives the conclusion of this paper.

2. Related work and background

There are three primitive methods of DDoS attack detection in SDN: the method based on statistics, the method based on policy and the method based on machine learning.

In SDN network, the statistical-based method refers to the usage of statistical data to distinguish malicious traffic and normal traffic. The common method of DDoS detection is to measure the statistical characteristics of networking traffic flow. Generally, entropy variations of selected packet fields should be monitored. Entropy, by definition, is the degree of chaos in the data, or a measure of the randomness of the data. In DDoS attacks, the randomness of networking flows characteristics will be affected by sudden changes. The typical characteristic of the DDoS attack is that a great deal of attackers usually utilize the compromised device to send a lot of communication to one or more end hosts. Therefore, traditional DDoS attacks usually lead to an increase in the entropy value of the source IP address or a decrease in the entropy value of the destination IP address.

Mousavi et al. [9] leveraged the entropy of the target IP address to detect DDoS attacks. When the switch received a packet and did not know what to do with it, it sent a Packet-in message to the SDN controller. Packet-in messages contained the target IP address, and the controller calculated the entropy of the target IP address. Set the sampling window size and threshold in the controller. When the calculated entropy value was less than the set threshold value, it was determined as a DDoS attack. Wang et al. [10] processed the statistical information of switch flow table in SDN network to detect DDoS attacks. Kiibra Kalkan et al. [11] used flow-based detection mode when detecting normal traffic. Packet-based traffic detection mode was used when detecting malicious traffic. The scheme in this paper could drop malicious packets after detecting the attack packets, so as to achieve effective defense effect.

Although DDoS attack detection methods based on information entropy occupy less resources, these detection methods all need to choose the appropriate threshold to get the desired detection results. Because of the different traffic types in different networks, it is quite a challenge to determine the proper detection threshold in different attack environments.

In a policy-based detection scheme, if the flows detected conforms to a specific policy, the network flow examined is considered normal. Conversely, the network flows being examined is considered malicious [12]. Shin et al. [13] proposed a strategy, which applied TCP connection status information to detect and mitigate DDoS attacks. In this scheme, Avant-guard detected and classified normal and abnormal flows in the system. Wei et al. [14] proposed another strategy-based anomaly detection system. The system calculated the trust value of all sources. If the value was less than the threshold set by the system, it was determined that these sources were attackers. The system further removed packets sent by these sources to mitigate DDoS attacks. The attack detection method based on policy has higher accuracy. However, it needs to redeploy the policy when encountering a new attack pattern.

The machine learning algorithms can grasp the characteristics of samples through learning, and finally form a very accurate model for detecting such features through training. Ravi et al. pointed out in [15] that the application of machine learning in network anomaly detection had been extensively studied. Jin Ye et al. [16] extracted the six tuples in the flow tables of the switches as feature vectors, which were as the input of SVM algorithm to detect malicious traffic. Peng et al. [17] took advantage of the relevant information of the data set as the detection feature of K-nearest Traffic classification and correlation analysis (CKNN), so as to classify the normal and abnormal traffic. Trung et al. [18] combined SVM and SOM algorithm to detect DDoS attacks. The SVM was used to detect the network flow in the first step, and the network flow that cannot be processed in the first step was detected by the SOM algorithm.

Recent research have shown that separating the control and data plane, centralized control of SDN and dynamic access of network equipment can be achieved the support IoT devices. A feasible solution to reinforce the security control and management capability of the IoT is to combine with SDN [19]. The paper [20] proposed a software-defined IoT security service framework, using SDN security controller to provide effective privacy protection, access control, key management and other security services. The paper [21] proposed an IoT architecture model, which combined the pros of centralized control of SDN and computing of fog. The paper [22] proposed a novel SDN controller design scheme based on the IoT multi-network. The paper [23] divided the IoT into different subnets, and implemented a cross-domain secure routing strategy using a software-defined security cluster head. The above research results prove the feasibility and effectiveness of the combination of SDN and the IoT.

In the SDN framework, how to improve the security of the IoT is an important research topic. Nobakht et al. [24] proposed a framework called IoT-IDM, which was based on the SDN with IoT attack detection and mitigation mechanism. It could detect the target host and mitigate the attack. Salman et al. [25] proposed a resultful resolution for IoT security, which was the identity authentication scheme combining SDN and IoT. Gonzalez et al. [23] proposed an IoT security scheme based on SDN applicable to distributed cluster routing protocol. The author built a test platform for the proposed security solution to verify and evaluate it. Nguyen et al. [26] analyzed specific vulnerabilities of link spoofed attacks on link services in SDIoT networks by using controllers, and proposed a hybrid strategy to solve this security problem. Bull et al. [27] proposed a security system based on flow for equipment in the IoT, which aimed to reduce the damage caused by DDoS attacks. In order to improve the accuracy of sample-based anomaly detection in the IoT, SDN was introduced into the system [28].

The IoT is a technology that can link up anything and anywhere.Security in the context of IoT is a critical challenge [29]. Specially, numerous challenges prevent the securing of IoT devices and their end-to-end communication in an IoT environment [30]. In the IoT ecosystem, all connected objects can be connected to the Internet and all have a distinctive identifier, which is used for addressing. The IoT has various forms, complex technologies and widespread implications. From up to down, the IoT can be divided into comprehensive application layers, management service layers, network construction layers and perception and recognition layers, respectively in accordance with the principles of application, processing, transmission and information generation. As the name implies, the comprehensive application layer is a set of various applications. The number of network applications has surged, showing the characteristics of diversification, scale, and industrialization [31]. With the aid of vast storage technology and the management service layer of high-performance computation, which provides the intelligent development platform for the application of the upper layer. It is the main function of the network construction layer to link the perceptual recognition layer devices to the Internet. The essential technology of the IoT is perception and identification. Perception and identification layer contains automatic information generation equipment such as radio frequency identification and wireless sensors, as well as diverse smart electrical products such as our common computers. The essential technology of the IoT is perception and identification. Perception and identification layer contains automatic information generation equipment such as radio frequency identification and wireless sensors, as well as diverse smart electrical products such as our common computers [32]. Because of the numerous devices and applications, the security of the IoT has been threatened unprecedentedly.

3. DDoS attacks in proposed SD-IoT framework

Our proposed SD-IoT architecture is an expanded version of SDN combined with the IoT. Three layers, the infrastructure layer, the control layer and the application layer, are included in the framework. The proposed architecture is shown in Figure 1.

The location of the network equipment is the infrastructure layer, which includes many switches. our scheme is to fuse SDN switches and IoT gateways into SD-IoT switches, that is, SD-IoT switches have both SDN switch functions and gateway functions. Our solution is different from that in literature [33], that the IoT gateways and SDN switches in this framework are independent. IoT drivers and sensor equipment such as personal computers, digital cameras, and smart phones can be connected to our SD-IoT switches.

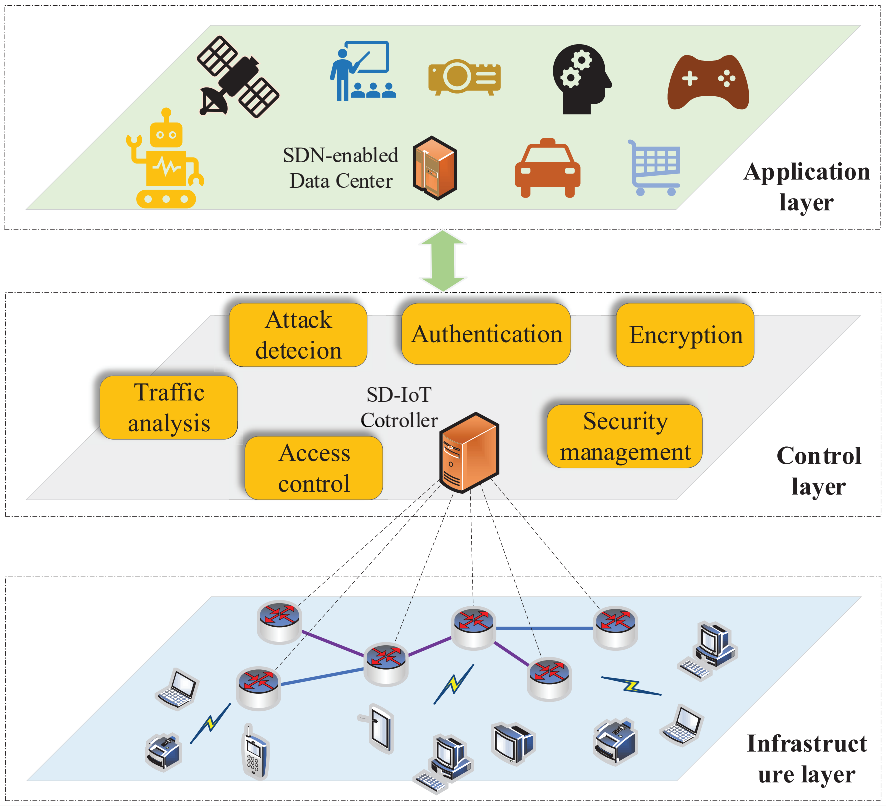

Figure 1. The SD-IoT framework.

The SD-IoT controller is the component of the control layer. The SD-IoT controller utilizes the southbound interface to collect topology information of IoT devices, establish a global view, and then complete network management functions such as attack detection, traffic engineering and load balancing on the infrastructure layer. At the same time, this layer also provides the corresponding API for the application layer to invoke.

The application layer in this framework involves of a variety of applications running in the IoT server and connect to the SD-IoT controller through a northbound interface. At the same time, it also provides convenience for developers. In SD-IoT, developers no longer worry about differences in underlying device communication protocols by using a unified south interface protocol, which simplifies application development, facilitates application deployment and reduces network maintenance costs.

In our framework, the centralized logical control of IoT devices is the responsibility of the SD-IoT controller. The benefits of logical centralized control are configuration and management, but the drawbacks are also obvious, namely, the system can be exposed to danger. Our proposed programmable SD-IoT framework resemble SDN, which plays a constructive effect on detecting DDoS attacks. As shown in Figure 2, the DDoS attack process in our proposed framework is analyzed as follows.

a) Both normal users and DDoS attackers send packets to the SD-IoT switch. Normal packets are retrieved from daily traffic and attack packets are produced by attack scripts.

b) The SD-IoT controller periodically issues instructions to collect information about the current packets header of the SD-IoT switch.

c) The SD-IoT switches proceed to the next step based on the result of SD-IoT controller.

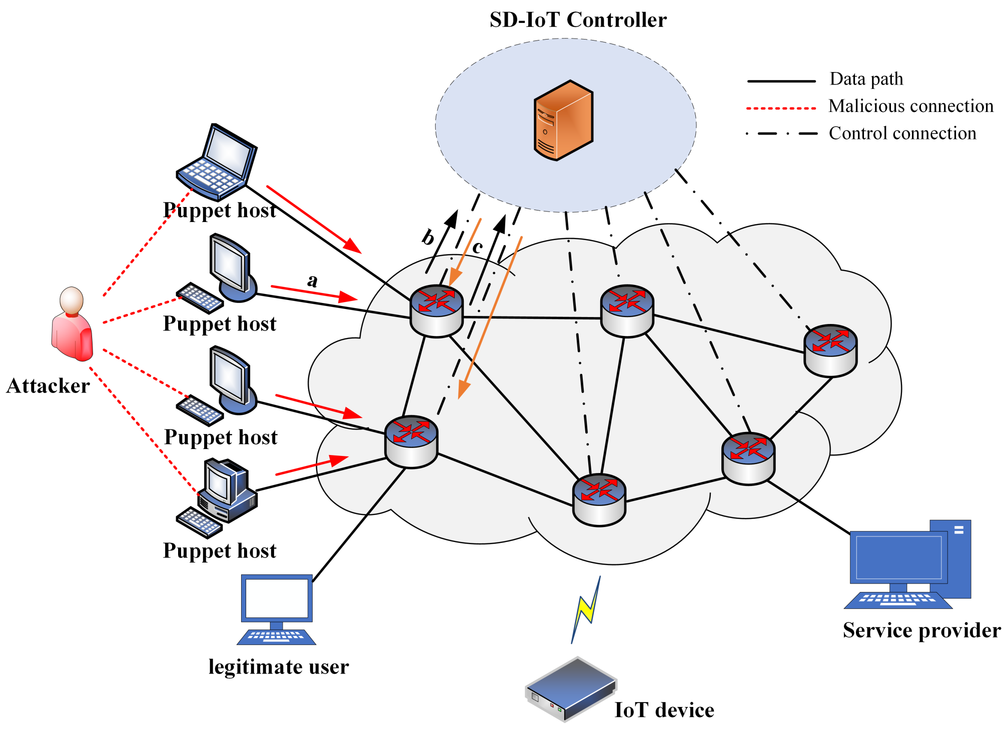

Figure 2. DDoS attacks in SD-IoT.

4. DDoS attack detection model

The model IFACNN can detect DDoS attacks in SD-IoT efficiently and accurately. The four components of this model are introduced below.

4.1. Data packets collection and data preprocessing model

Data Packets collection: The foremost purpose of this module is to collect the header information of the current data packets of SD-IoT switches periodically. Take advantage of the programmability of the architecture, write a program module at the SD-IoT controller to send instructions to the switch to collect packets. The time interval at which the controller collects the packet header from the switch is extremely important. If the collection time interval is relatively long, the system may be unable to respond in time when it is subjected to a DDoS attack, resulting in irreparable loss and damage. Conversely, if the collection interval is too short, the number of interactions between the controller and the switch will enhance, increasing the overhead of processing data for the SD-IoT controller. Considering the above factors, the SD-IoT controller in this system obtains data packets header information from the SD-IoT switches every 6 seconds.

Algorithm 1 Converting Packets into Networking Flow Algorithm

Input:
pkt.tuple

Output:
The index value of the networking flow

```
1: Procedure GetIndex ($pkt.tuple$)
2: $T \leftarrow 0$ ▷ Initialise the networking flow
3: $\varepsilon \leftarrow 0$
4: for all $pkt \in NDP$ do
5:     for $idx$ init 0 to $T.length()$ by 1 do ▷ Loop over the networking flow
6:         if $pkt.tuple == T[idx]$ then
7:             break
8:             if $idx == T.length()$ then ▷ The networking flow was not found
9:                 $T.append(pkt.tuple)$ ▷ Add a new networking flow
10:                $\varepsilon[idx].append(pkt.header)$
11: End procedure
```

Table 1. Glossary of symbols.

Notation	Definition
NDP	The networking data packets set
idx	The index
fid	The index of networking flow
ε	A collection of packets sorted by networking flow
M	The sub-flow window set
m	The sub-flow window
V	The feature vector
T	The networking flow according to the 5-tuple
L	The lable of networking flow
h	The number of networking flow

Data packets preprocessing: Networking flow refers to the sequence of data packets with the identical multiple tuples. For example, we define that data packets in the network are the same network flow when they own the same {Source IP, Source Port, Destination IP, Destination Port, Protocol} information. $S = \left\{p^{(1)}, p^{(2)}, ..., p^{(l)}, ..., p^{(n)}\right\}$ can be utilized to describe a network flow with n packets, where $p^{(l)}(1 \leq l \leq n)$ stands for the l-th packet of S. The q-vector containing the information $p^{(l)} = \left\{p_1^{(l)}, p_2^{(l)}, ..., p_m^{(l)}\right\}$ can be represented as $p^{(l)} \in \mathbb{R}^q$. For instance, the source IP address, destination IP address and other information in the packet header. After we obtain the packets header, we divide the data packets into different network flows according to the quintuple. Algorithm 1 describes the detailed process. In Algorithm 1, the two for loops are nested and their time complexity is $o(n^2)$. Since the space required for the implementation of this algorithm does not change with the size of a certain variable, the space complexity of this algorithm is a constant, which can be expressed as $o(1)$. The symbols are defined in Table 1.

Algorithm 2 Data preprocessing algorithm

Input:
Networking traffic flow

Output:
List of labelled samples

```
1:  **Procedure GetIndex** ($pkt.tuple$)
2:  $M \leftarrow 0$ ▷ Initialise the sub-flow window set
3:  $fid \leftarrow 0$
4:  $V \leftarrow 0$
5:  while $fid < T.length()$ do
6:      $m \leftarrow 0$
7:      $m_{row} \leftarrow 0$
8:      $m_{col} \leftarrow 0$
9:      while $m.length() < n$ do
10:         $m[m_{row}, m_{col}] \leftarrow \varepsilon[fid].shift()$ ▷ Move $\varepsilon[fid]$ data into m
11:         if $true == \varepsilon[fid].empty$ then ▷ When the current $\varepsilon[fid]$ is empty, go to the next networking flow table
12:             $fid \leftarrow fid + 1$
13:             $m_{col} \leftarrow 0$
14:             $m_{row} \leftarrow m_{row} + 1$
15:             if $m_{row} \geq h$ then
16:                 break
17:     $M.append(m)$ ▷ Fill in M with m
18: for all $m \in M$ do
19:     $V.append(m.features)$
20:     $V.lable \leftarrow L.[m]$ ▷ Apply the lable
21: End procedure
```

We regard all the data packets in the network flow, assuming that there are A data packets. In these A packets, n packets are taken as a packets window, and there are h sub-flows in a packets window. In the following experiments, we chose to use 250 packets as a window. When $h < x$, we extract 6 features of each networking traffic flow in the window, and the remaining $x - h$ traffic flow features will be filled with zeros as the input to the IFACNN model. If $h > x$, each h flow features constitutes a set of inputs to IFACNN. As shown in Algorithm 2. In Algorithm 2, the two while loops are nested, and the for loop and while loop are parallel, so their time complexity is $o(n^2)$. Like the space complexity of Algorithm 1, the space complexity of this algorithm is also $o(1)$. The influence of the size of the h value on the detection rate will be considered in Section 5.

4.2. Feature extraction module

The core function of this module is to obtain the characteristics of network flow through data packets collection and data packets preprocessing module. We require determining the characteristics of the network flow as the input of the deep learning algorithm for DDoS attack detection. Although DDoS attackers can make use of a variety of attack methods, most attacks network flows are subject to certain rules. Hence, we can utilize network flow characteristics to detect DDoS attack. Based on the previous analysis, we have selected six characteristics associated with DDoS attacks.

The number of packets per networking flow (NPf): It is found that there is a significant difference in the number of packets between DDoS attack networking flow and normal networking flow. The attackers exploit the randomly produced fake source IP addresses to attack the victim host. Although this method can generate a great deal of networking flows within a short time, the characteristics of the attack networking flows are also obvious, that is, the number of packets per attack networking flow is merely 1–3. Normal networking flow possess extensive packets. For that reason, we apply NPf to indicate one of the networking flow features.

The number of bytes per networking flow (NBf): For the sake of sending a large quantity of packets at short notice, attackers need to make the number of bytes of each attack networking flow extraordinary tiny. For example, the attackers send 120 bytes TCP flood attack packets to attack the victim host, while the number of bytes in a normal networking flow is much larger than this number. Thus, a significant feature for detection DDoS attacks is NBf.

Duration of each networking flow (DNf): The duration of the normal networking flow is long. The abnormal networking flow is plenty of useless packets randomly sent by the attacker with different source IP addresses, so the duration of the abnormal networking flow is short.

Rate of networking flow (RNf): Since DDoS attackers can send mountains of unusable networking flows, the available resources of the victim host will be occupied. Consequently, during an attack on a victim host, a good supply of associated networking flows increases dramatically. Therefore, RNf is also an important feature of DDoS attack.

Source IP addresses of networking flow (SIP): As previously mentioned, DDoS attackers send massive packets of fabricate source IP addresses to the victim host. For the attack networking flow where the attacked host is the destination address, the source IP addresses are fairly scattered and possess great randomness in a packets window which is predefined.

Destination IP addresses of networking flow (DIP): The destination IP addresses of the DDoS attack networking flow is extra focused and less random than that of the benign networking flow.

4.3. Improved firefly algorithm

Firefly algorithm (FA) is a kind of bionic swarm intelligence optimization method. Fireflies have two elements: brightness and attractiveness. Fireflies with strong luminescence can attract fireflies with weak luminescence. The iteration of position is completed in the process of moving from the weak firefly to the strong firefly. According to the above principle, the search for the optimal solution can be realized in the step-by-step iterative process.

The FA is a population-based algorithm, in which each firefly is represented as a vector point in the search space, that is, a solution to the problem. The candidate solution c_x can be expressed as the position of firefly $x : C = (c_{x1}, ..., c_{im})$, where $x = 1, 2, ..., \xi$, ξ represents the total number of individuals in the firefly population, and m represents the dimension of the problem. The relative brightness and relative attraction of fireflies can be expressed as the following equation:

$$\theta = \theta_0 e^{-L\delta_{xy}} \tag{4.1}$$

$$\mu(\delta) = \mu_0 e^{-L\delta_{xy}^2} \tag{4.2}$$

where, θ_0 represents the fluorescence brightness of firefly at $\delta = 0; \mu_0$ is the maximum degree of attraction, that is, the degree of attraction at $\delta = 0$; L is the parameter of light absorption, which represents the characteristic that the fluorescence is weakened by the influence of distance and propagation medium, and can be set as a constant; δ_{xy} represents the distance between position x and position y. The Euclidean distance between two fireflies can be expressed as:

$$\delta_{xy} = \| c_x - c_y \| = \sqrt{\sum_{k=1}^{m} (c_{xk} - c_{yk})^2} \tag{4.3}$$

when fireflies are attracted to move, the distance between them will gradually shorten. According to the principle of equivalent infinitesimal substitution, Eq (4.4) can be used to replace Eq (4.2), so as to reduce the amount of calculation and improve the operation speed.

$$\mu(\delta) = \frac{\mu_0}{1 + L\delta_{xy}^2} \tag{4.4}$$

It is further concluded that the update iteration formula for the firefly to move from position x to another more attractive position y is as follows:

$$\begin{aligned} {c_x}^{n+1} &= {c_x}^n + \mu_0 e^{-L\delta_{xy}^2}(c_x - c_y) + \varphi(r - 0.5) \\ &= {c_x}^n + \frac{\mu_0}{1 + L\delta_{xy}^2}(c_x - c_y) + \varphi(r - 0.5) \end{aligned} \tag{4.5}$$

where φ is the step size factor, n is the current number of iterations, r is the random factor between [0,1], which follows a uniform distribution.

Aiming at the problems of slow convergence speed and easy to fall into local optimal value of FA, we introduce a location update strategy based on population diversity. In order to avoid the vibration problem of the optimal solution and improve the accuracy of optimization, we propose an adaptive step size update measure.

According to Eq (4.5), when each firefly updates its position, it mainly depends on the attraction of fireflies with high fluorescence brightness to itself. The analysis shows that due to the lack of randomness of global search, the whole search space may converge to the local optimum after several location updates. Therefore, this paper proposes to use the average distance length from individual to group center to measure group diversity firstly, as shown in Eq (4.6):

$$\beta^n = \frac{1}{|S|} \sum_{x=1}^{|s|} \sqrt{\sum_{y=1}^{m} (c_{xy} - \overline{c_Y})^2} \tag{4.6}$$

where, β^n represents the diversity index of the n-th generation population, $|S|$ represents the population size, and $\overline{c_Y}$ represents the j-dimensional component of the average center of the population.

Based on the diversity characteristics introduced above, the strategy of firefly location update is further adjusted. The modified location update formula is as follows:

$${c_x}^{n+1} = {c_x}^{n} + \mu_0 e^{-L\delta_{xy}^2}(c_x - c_y) + \rho * (c_x - c_b) + \varphi(r - 0.5) \tag{4.7}$$

where, c_b is the current optimal firefly individual, ρ is a weight that changes according to population diversity and iteration times and has a certain randomness. The calculation formula of ρ is as follows Eq (4.8):

$$\rho = \{ \begin{matrix} -r*\sigma & \beta^n \leq \sigma*\beta^0 \\ 0 & \beta^n > \sigma*\beta^0 \end{matrix} \tag{4.8}$$

where, β^0 represents the population diversity index at the initial time; σ is a linear decreasing function, which decreases with the increase of the number of iterations. The calculation formula of σ is as follows Eq (4.9), in which $T_{\max}$ is the maximum number of iterations and T_{itex} is the current number of iterations.

$$\sigma = \frac{T_{\max} - T_{itex}}{T_{\max}} \tag{4.9}$$

According to Eq (4.7), in the initial stage of the algorithm, the value of σ is relatively high, and the result obtained by Eq (4.8) is a negative value. Therefore, the probability of firefly individual will move towards the irregular direction away from the best in the initial stage, so as to ensure the search in a wider range; while in the later stage of the algorithm, due to the reduction of diversity requirements, the firefly will search randomly in the direction close to the best to achieve more refined local search. Such a dynamic adjustment mechanism can balance the problems of global optimization in the early stage and local optimization in the later stage, so as to better avoid the phenomenon of convergence of the algorithm to local optimization.

In the FA, the step factor φ in Eq (4.5) is a fixed value. In the later stage of the algorithm, most fireflies will do local optimal search around the best point, and fixed step size may lead to failure to converge to the best point. Therefore, in order to balance the contradiction between the early and late stages of the algorithm convergence process, the step size factor is dynamically adjusted according to the number of iterations as follows:

$$\varphi = \varphi_0 * \sigma \tag{4.10}$$

where, φ_0 is the initial step factor. So far, we have completed the improvement of firefly algorithm.

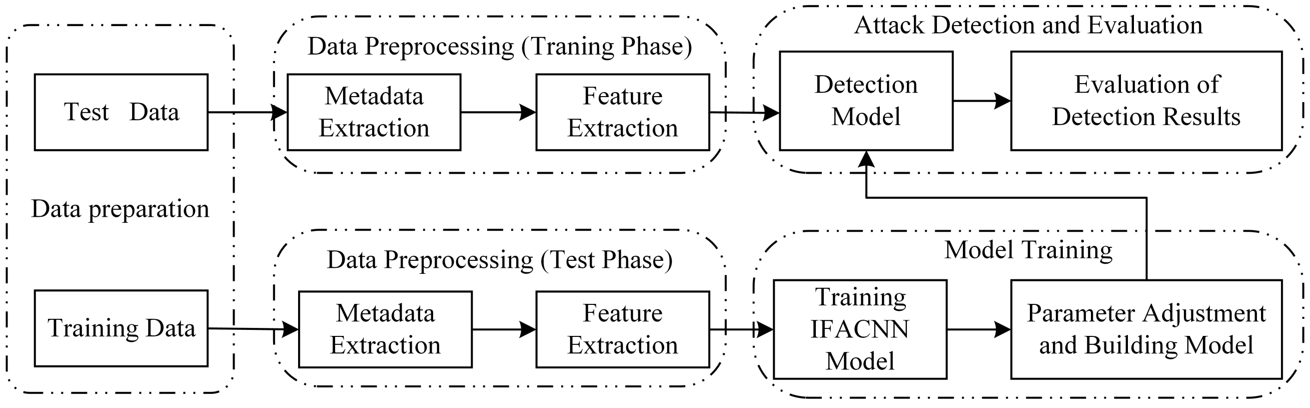
Data preparation
Test Data
Training Data
Data Preprocessing (Traning Phase)
Metadata Extraction
Feature Extraction
Data Preprocessing (Test Phase)
Metadata Extraction
Feature Extraction
Attack Detection and Evaluation
Detection Model
Evaluation of Detection Results
Model Training
Training IFACNN Model
Parameter Adjustment and Building Model

Figure 3. Flowchart of DDoS attack detection.

4.4. DDoS attack detection

For the reason that the characteristics of attack networking flows and normal networking flows are distinctive, attack detection is considered as a classification problem. Data packets collection and data packets preprocessing module collects data packets and preprocesses the data. The features of networking flows are extracted by feature extraction module. Ultimately, train the attack detection module through the data samples. Figure 3 shows the process of DDoS attack detected by IFACNN. The training of IFACNN neural network should gradually adjust and reduce the error between network output and expectation by constantly adjusting the weight and bias between each layer, and finally achieve the target accuracy requirements. The topology structure of each layer and the learning rate in IFACNN network are all part of the hyperparameters of neural network. The adjustment of hyperparameters has great influence on the final network performance and detection accuracy of neural network. We use the improved firefly algorithm to optimize the selection of IFACNN hyperparameters. Each firefly individual represents the hyperparameter configuration of a neural network. During training, the training accuracy of the neural network is measured by the root mean square.The trained IFACNN model is exploited to detect the DDoS attack on the networking flows of the system and export the detection results.

Deep learning model: The IFACNN model is mainly composed of input layer, convolution layer, pool layer, fully connected layer and output layer. It is the function of the input layer to accept the data to be detected. The convolution layer is used to take local features, and full connected layer is to assemble the previous local features into a complete graph through the weight matrix. The function of pooling layer is to remove redundant information, compress features and simplify network complexity.

Input layer: As mentioned above, data packets are divided into different networking flows according to five tuples, and different features are extracted from the networking flows. In the predefined networking flow window, the networking flows and features constitute a two-dimensional input matrix. The IFACNN algorithm studies the correlation of networking flows in that window. We can represent the flow of the window as F matrix, which size is $h \times f$. For example, $F = \{flow_1, flow_2, ...flow_h\}$ is the matrix in a window of our model, where F has h flows and every flow owns $f = 6$ characteristics, as shown in Figure 4.

CNN layer: Filters is also known as kernels or sliding windows. The input matrix F is convolved with e filters, each of which has a size of $h \times i$, where h is the length of the filter and i is the width of the filter. Through convolution operation, the algorithm gets and studies useful partial features for normal flow and DDoS attack classification. One convolution yields an activation map c of size $f - i + 1$, as follows

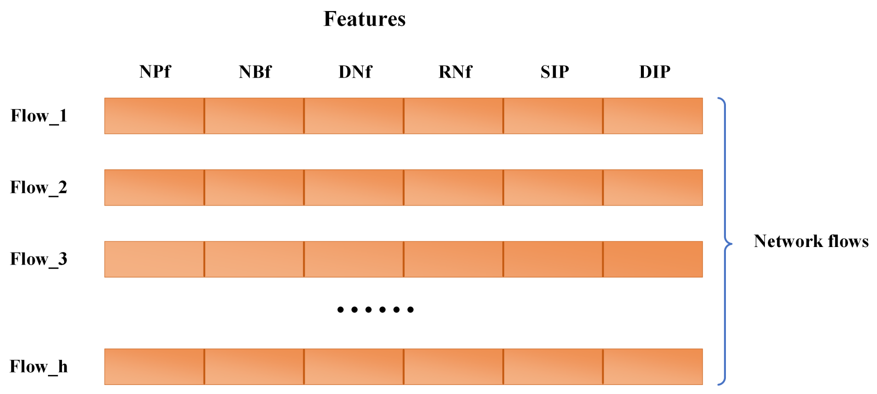

Figure 4. The input of CNN model.

$$c_e = \mathrm{Re}LU(Conv(F)\, W_e, b_e) \tag{4.11}$$

The weight of the e-th filter learned in the model training stage is W_e, and the bias parameter is b_e. With the intention of overcoming the problem of gradient disappearance and accelerate the training speed, we introduced the linear rectifier function ReLU:

$$\mathrm{Re}LU(y) = \max(0, y) \tag{4.12}$$

Stack all activation maps to construct an activation matrix C of size $((f - i + 1) \times e$, so that $C = [c_1 \,|\, c_2 \,|...|\, c_e]$.

Max pooling layer: We adopt the under sampling for C. The output matrix q_0 of size $\{[(f - i + 1)/q] \times e\}$ is generated by a pool of size q. Among them, each learning filter possesses the hugest q activation, making $q_0 = [\max(c_1)|...|\max(c_e)]$. This approach allows our model to focus on the larger activation and ignore some of the less valuable information. This also indicates that pooling layer can compress information characteristic encoding, thus decreasing the complication of the network. Then the output matrix is flattened, and the one-dimensional data is input into the classification layer.

Classification layer: Input the output data of the pooling layer to the full connection layer. Our output layer consists of two nodes, and deliver the output data x to the activation function sigmoid:

$$\sigma(y) = 1/(1 + e^{-y}) \tag{4.13}$$

The output value from the activation function is between 0 and 1. therefore returning the possibility that given networking flows is a malicious DDoS attack $p \in [0, 1]$. When $p > 0.5$, we classified these networking flows as DDoS attack flow, while other p value networking flows are classified as normal flows.

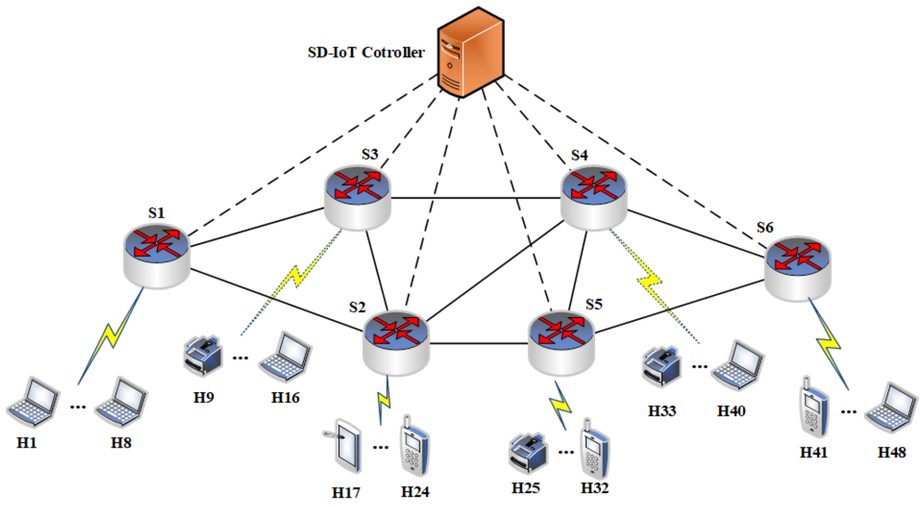

Figure 5. The topology of DDoS attack detection for SD-IoT.

5. Performance evaluation

To begin with we configure the experimental settings in this section, and then we estimate the performance of the proposed algorithm.

5.1. Simulation setting

We take advantage of SDN-WISE-CONTIKI and mininet to simulate the real SD-IoT environment. All the software is deployed on the Windows 10 Virtual-Box platform, and the operating system Ubuntu 16.0.4LTS. In particular, the SD-IoT controller in the system utilizes floodlight of an Open source controller, while the SD-IoT switch exploits Open vSwitch. Deep learning module is developed based on Tensorflow framework. At the same time, SDN-WISE-CONTIKI is applied to create Sink nodes of sdn-wise sink type and sensing nodes of sdn-wise mode type. Additionally, normal packets and DDoS attack packets in the system are created through Scapy of the python script, corresponding to typical network attack packets such as ICMP packets, UDP packets and TCP packets.

Figure 5 presents the SD-IoT network topology generated in Mininet, and we employ the topology to verify the proposed DDoS attack detection mechanism. Our topology consists of a SD-IoT controller implemented by floodlight, and six Open vSwitches S1, S2, ...and S6 are used as SD-IoT switches. The IoT embraces 48 terminal equipment, which are entitled H1, H2, ... and H48, including host, wireless nodes and so forth. The three hosts H1, H3 and H5 connected to S1 are selected as the attack hosts, which send a large number of fake source IP address packets to the victim host H33 through Scapy script to launch attacks on SDN network.

The detection performance of the IFACNN model is evaluated by five evaluation indexes, namely F1 Score, Recall, Precision, and Confusion Matrix. True positive (TP) refers to the quantity of data that the detection model judges normal data to be normal; False positive (FP) refers to the quantity of data that the detection model identifies DDoS attack data as normal data; True negative (TN) refers to the quantity of DDoS attack data correctly identifies by the detection model as attack data; False negative (FN) refers to the quantity of data that the detection model determines normal data to be a

Table 2. Number of samples.

Dataset	Normal samples	DDoS samples
Training Set	168948	128928
Test Set	112633	85953

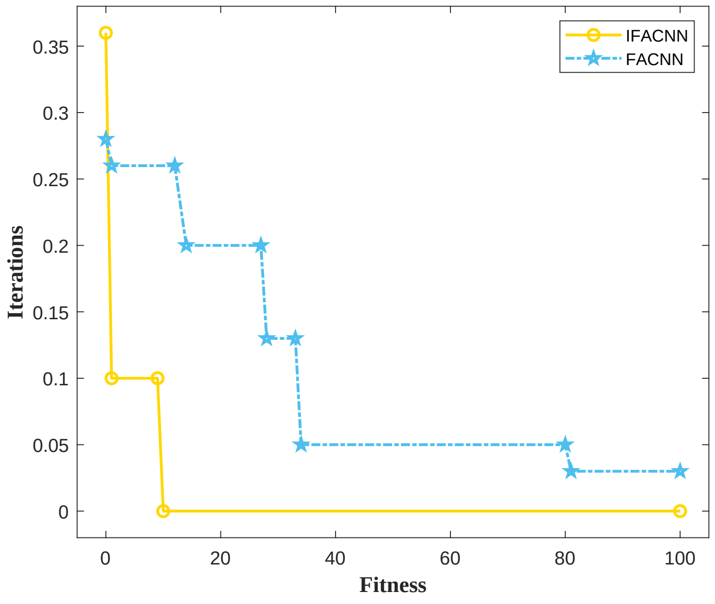

Figure 6. Improved firefly fitness.

DDoS attack.

Accuracy represents the percentage of data classified correctly by the detection model to the total data:

$$Accuracy(Acc) = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

Precision refers to the percentage of the data packets that the model determines to be attack that are actually the number of attack packets:

$$\Pr ecision(\Pr) = \frac{TP}{TP + FP} \tag{5.2}$$

Recall denotes the percentage of all DDoS attack data estimated by the detection model as DDoS attack data:

$$\mathrm{Re}call(Rc) = \frac{TP}{TP + FN} \tag{5.3}$$

The harmonic mean of recall and precision signifies F1 score, which could assess detection model implementation more appropriately.:

$$F1 = \frac{2 * \Pr ecision * \mathrm{Re}call}{\Pr ecision + \mathrm{Re}call} \tag{5.4}$$

The confusion matrix is mainly intended for the classification results of detection model and the degree to which the data matches the actual labels.

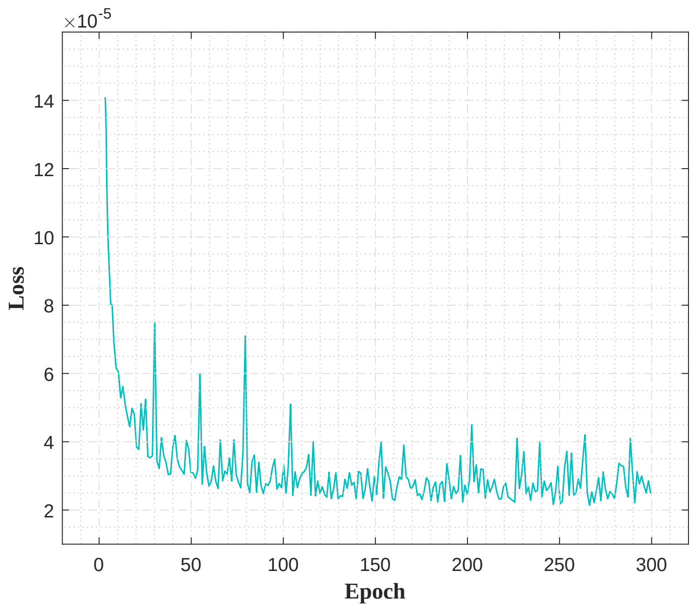

Figure 7. Loss function.

5.2. Evaluation of IFACNN model experimental results

There are 496462 experimental data packets that come from the real SDN network, including 281581 normal packets and 214881 DDoS attack packets and 60% of which are used for training and the rest for testing. As shown in Table 2.

A. Fitness and Loss: The output dimension is two in CNN network with improved firefly algorithm. The input feature dimension and the node number of each layer in the hidden layer are optimized as firefly features. In the SD-IoT network, we set the population size of fireflies as N = 50, the maximum iteration times as 100, and the initial step factor as 0.6. The fitness changes in the optimization process of firefly algorithm are shown in Figure 6. As can be seen from Figure 6, the improved firefly converges to the optimal fitness of 0.0001 around 10th generations, and the algorithm tends to be stable. Through the improvement of the algorithm, the speed of finding the ideal network structure and learning rate of CNN is accelerated.

The obtained parameters are used to construct the CNN detection model, and the training data is used to train 300 generations. The loss curve of the model is shown in Figure 7. It can be seen that since generation 100, the loss of detection model is basically stable at about 0.000025.

B. Different depths of convolution layers: The detection accuracy of a well-behaved IFACNN model is affected by many factors, and the depth of convolutional layer is extremely important. We build four IFACNN models with different kinds of depths. Parameters selection for each single IFACNN model are in Table 3. Among them, 3C2P2F indicates that the model contains 3 convolution layers, 2 maximum pooling layers, and 2 fully connection layers.

In this paper, four evaluation indexes are used to evaluate the performance of the four models mentioned above. They are precision, accuracy, recall and F1. Through the experiments and comparative analysis,the evaluation results are shown in Figure 8.

Table 3. Parameters selection for each single IFACNN model.

IFACNN-Parameters	IFACNN model			
Layers	$2C2P2F$	$2C2P3F$	$3C2P2F$	$3C3P2F$
Activation Function		$Relu, Sigmoid$		
Population Size		50		
Maximum Iteration Times		100		
Initial Step Factor		0.6		

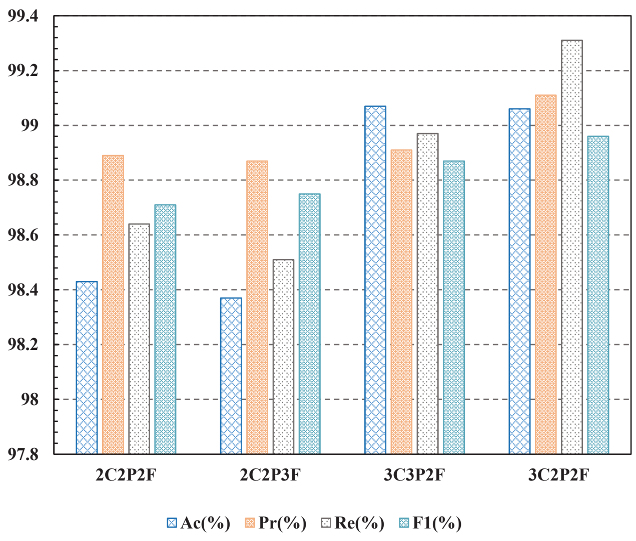

Figure 8. Performance comparison of different layers.

From the picture we can see that detection results of models shown are all higher that 98%. This shows that the CNN model has strong superiority in detecting DDoS attacks. Our model (3C3P2F, 3C2P2F) with 3 convolution layers is obviously pass beyond that with 2 convolution layers (2C2P2F, 2C2P3F). Compared with 3C3P2F model, 3C2P2F model has higher recall rate, accuracy and F1 scores, but the accuracy rate is marginal lower.

We leverage confusion matrix to further analyze the detection accuracy of the four models and the results are shown in Figure 9. Figures a, b, c, d are 2C2P2F, 2C2P3F, 3C3P2F and 3C2P2F respectively. The results show that the three-layer convolution model has better detection ability than the two-layer convolution model. In addition, the detection accuracy of 3C2P2F model (0.99) is the highest compare to others. Considering the importance of accuracy to DDoS detection system and the complexity of CNN model, the 3C2P2F model is evaluated as the best.

C. The effect of various h values: In order to detect DDoS attack on networking flows, our detection method requires to define h networking flows in predefined packet windows. We investigate the effect of various h values on performance in our experiment. We select different h values, such as h = 5, 10, 15, 20, 25 and 30 respectively, to train the proposed model.

Through the analysis of network data, we recognize that the traffic flow with a good deal of packets is generally normal networking flow. IFACNN model has apprehended these characteristics in the process of training, and then made the corresponding determination.

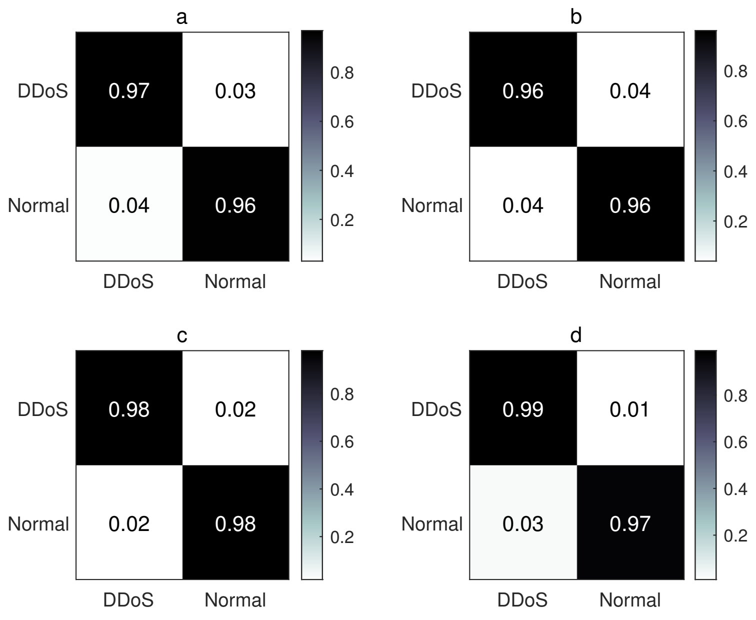

Figure 9. Confusion matrix of 4 CNN models.

Table 4. The effect of various h values.

The value of h	Ave*Ac	Ave*Pr	Ave*Rc	Ave*F1
5	99.12	99.23	99.46	99.11
10	99.18	99.20	99.41	99.04
15	99.09	99.17	99.29	98.99
20	99.06	99.11	99.31	98.96
25	97.91	97.96	97.25	96.12
30	97.75	96.67	96.42	94.93

Table 4, Figures 10 and 11 show the consequences for the proposed 3C2P2F IFACNN model. Ave are average values. As the table shown, with the value of h increasing, the model is allowed to examine more networking flows. As h increases, the performance of the model gradually decreases, but the magnitude of the decrease is different. In Table 4, when the value of h is 5, 10, 15, 20, their 4 average indicators almost reach 99%. When the value of h is 25, the average index value drops to 97%. Detecting 25 networking flows significantly reduce the performance of our scheme. In the predefined packet windows, the ones with less networking traffic will be padded with 0. That fillings can confuse the model and cause degradation in detection performance.

Figure 10 and Figure 11 show the training time and CPU usage of IFACNN algorithm under different H values. With the increase of h, the training time becomes shorter and shorter, and the utilization of CPU becomes larger and larger. When h is 5, the training time is 36.3 minutes. When h is 10, the training time is 27.6 minutes. When h is 15, the training time is 19.1 minutes. When h is 20, the training time drops to 9 minutes. When h is 25 and 30, the training time does not change significantly. That's because we have a certain entire quantity of samples data, and the fewer the number of networking flows in the data packet window, the number of training will take up a large part of the time. When

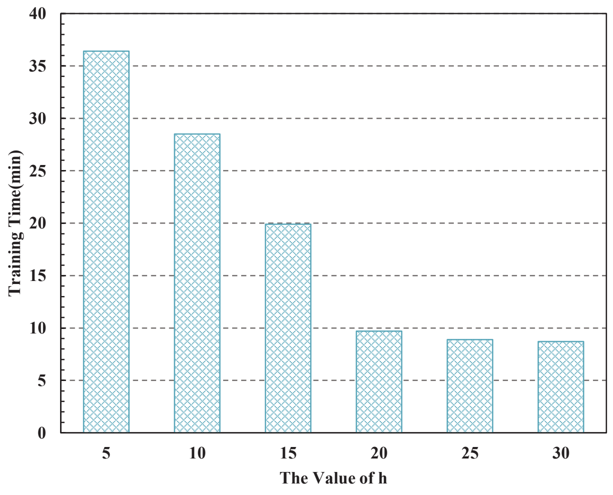

Figure 10. Training time of proposed IFACNN model.

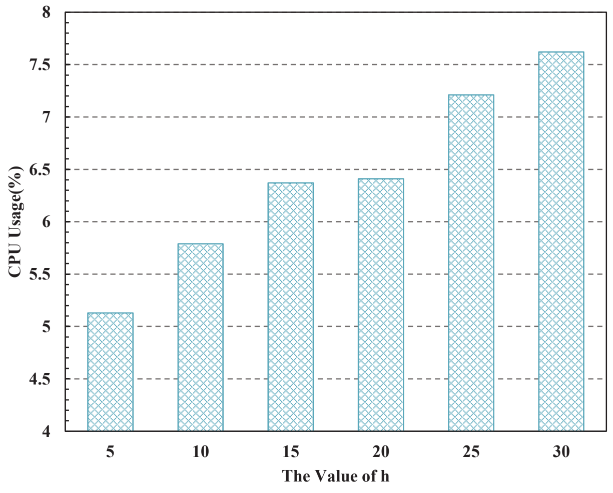

Figure 11. CPU usage of proposed IFACNN model.

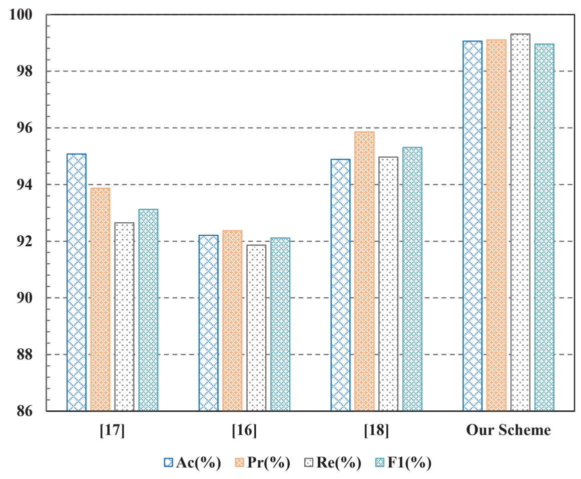

Figure 12. Performance comparison of different models.

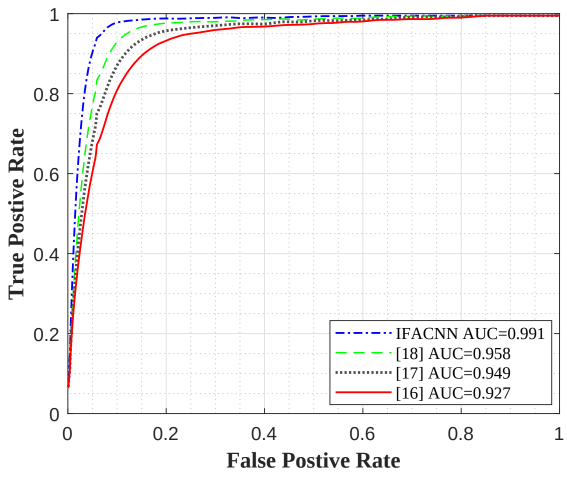

Figure 13. ROC in different machine learning.

the quantity of networking flows in the data packet window growths, the number of training decreases, and the training time also decreases. When h is 5, 10,15 respectively, the utilization rate of CPU rises from 5.1% to 6.4%. When h is 20, there is no significant change in CPU usage compared to when h is 15. When h is 25, CPU utilization increases again. Combined with Table 3, it can be concluded that when h is 20, IFACNN model has the best performance.

D. Compared to different machine learning: This paper continues to utilize above four evaluation factors to compare the performance of 3C2P2F model with traditional machine learning methods that mentioned in paper [16–18]. As Figure 12 illustrated, the DDoS attack detection rate of CNN algorithm is superior to traditional machine learning which detection rate is higher than 90%. It demonstrates the advantages of IFACNN in networking flows detection and possesses the capacity to differentiate DDoS traffic flows from normal traffic flows.

Furthermore, the area below the ROC (receiver operating characteristic) curves are calculated individually. Apparently, the larger the area, the higher the detection accuracy. The AUC value and efficiency of detection are positively correlated. As shown in Figure 13, the AUC of IFACNN method is 0.991, while the value of other machine learning methods is lower than IFACNN. Thus, the simulation shows that the IFACNN method can greatly reduce false negative rate of DDoS detection system.

6. Conclusions

DDoS attack is still one of the main threats to global network security. We design a general framework that consisting of SD-IoT controller, SD-IoT switch combined with IoT gateway and terminal IoT equipment. An algorithm that specialized in traffic preprocessing is put forward for this SD-IoT framework to detect DDoS attacks and to maximize the detection accuracy. Furthermore, We propose an improved firefly algorithm to optimize the neural network structure to improve the convergence time and detect accuracy. Meanwhile the 3C2P2F is verified as best IFACNN model by experiment and verification. Comparing to traditional machine learning from other papers, the IFACNN model that described above has advanced performance in terms of precision, recall, accuracy and F1. By compared different values of h, combining the detection capability, the training time and CPU usage, it can be concluded that when h is 20, the performance of the model is relatively superlative.

Acknowledgments

This was supported by Research on Key Technologies of Security Mechanism of Software Defined Internet of Things under Grant 2020YJS018 and the National Key Research and Development Program of China under Grant 2018YFB1800305.

Conflict of interest

The authors declare no potential conflict of interests.

References

1. I. Cvitić, D. Peraković, B. Gupta, K. K. R. Choo, Boosting-based DDoS detection in internet of things systems, *IEEE Int. Things J.*, **2021** (2021). doi: 10.1109/JIOT.2021.3090909.
2. F. Song, Z. Ai, H. Zhang, I. You, S. Li, Smart collaborative balancing for dependable network components in cyber-physical systems, *IEEE Trans. Ind. Inf.*, **17** (2021), 6916–6924. doi: 10.1109/TII.2020.3029766.
3. F. O. Catak, Two-layer malicious network flow detection system with sparse linear model based feature selection, *J. Nat. Sci. Found. Sri Lanka*, **46** (2018), 601–612. doi: 10.4038/jnsfsr.v46i4.8560.
4. CAICT, *White Paper on the Internet of Things*, 2020.
5. K. K. Karmakar, V. Varadharajan, S. Nepal, U. Tupakula, SDN-enabled secure IoT architecture, *IEEE Int. Things J.*, **8** (2021), 6549–6564. doi: 10.1109/JIOT.2020.3043740.
6. P. Mishra, A. Biswal, S. Garg, R. Lu, M. Tiwary, D. Puthal, et al., Software defined internet of things security: properties, state of the art, and future research, *IEEE Wireless Commun.*, **27** (2020), 10–16. doi: 10.1109/MWC.001.1900318.
7. F. O. Catak, A. F. Mustacoglu, Distributed denial of service attack detection using autoencoder and deep neural networks, *J. Intelli. Fuzzy Syst.*, **37** (2019), 3969–3979. doi: 10.3233/JIFS-190159.
8. F. Song, Y. Zhou, Y. Wang, T. Zhao, I. You, H. Zhang, Smart collaborative distribution for privacy enhancement in moving target defense, *Inf. Sci.*, **479** (2019), 593–606. doi: 10.1016/j.ins.2018.06.002.
9. S. M. Mousavi, M. St-Hilaire, Early detection of DDoS attacks against SDN controllers, in *2015 International Conference on Computing, Networking and Communications (ICNC)*, (2015), 77–81. doi: 10.1109/ICCNC.2015.7069319.

10. R. Wang, Z. Jia, L. Ju, An entropy-based distributed DDoS detection mechanism in software-defined networking, in *2015 IEEE Trustcom/BigDataSE/ISPA*, (2015), 310–317. doi: 10.1109/Trustcom.2015.389.

11. K. Kalkan, G. Gür, F. Alagöz, SDNScore: A statistical defense mechanism against DDoS attacks in SDN environment, in *2017 IEEE Symposium on Computers and Communications (ISCC)*, (2017), 669–675. doi: 10.1109/ISCC.2017.8024605.

12. N. Dayal, P. Maity, S. Srivastava, Z. Khondoker, Research trends in security and DDoS in SDN, *Secur. Commun. Networks*, **9** (2016), 6386–6411. doi: 10.1002/sec.1759.

13. S. Shin, V. Yegneswaran, P. Porras, G. Gu, AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks, in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, (2013), 413–424. doi: 10.1145/2508859.2516684.

14. L. Wei, C. Fung, FlowRanger: a request prioritizing algorithm for controller DoS attacks in software defined networks, in *2015 IEEE International Conference on Communications (ICC)*, (2015), 5254–5259. doi: 10.1109/ICC.2015.7249158.

15. N. Ravi, S. Mercy. Shalinie, Learning-driven detection and mitigation of DDoS Attack in IoT via SDN-cloud architecture, *IEEE Int. Things J.*, **7** (2020), 3559–3570. doi: 10.1109/JIOT.2020.2973176.

16. J. Ye, X. Cheng, Z. Jian, L. Feng, S. Ling, A DDoS attack detection method based on SVM in software defined network, *Secur. Commun. Networks*, **2018** (2018), 1–8. doi: 10.1155/2018/9804061.

17. P. Xiao, W. Y. Qu, H. Qi, Z. Y. Li, Detecting DDoS attacks against data center with correlation analysis, *Comput. Commun.*, **67** (2015). doi: 10.1016/j.comcom.2015.06.012.

18. V. Phan, N. Bao, M. Park, A novel hybrid flow-based handler with DDoS attacks in software-defined networking, in *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, (2016), 350–357. doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0069.

19. F. Song, M. Zhu, Y. Zhou, I. You, H. Zhang, Smart collaborative tracking for ubiquitous power IoT in edge-cloud interplay domain, *IEEE Int. Things J.*, **7** (2020), 6046–6055. doi: 10.1109/JIOT.2019.2958097.

20. F. I. Khan, S. Hameed, Software defined security service provisioning framework for internet of things, *Int. J. Adv. Comput. Sci. Appl.*, **7** (2017), 411–425. doi: 10.14569/IJACSA.2016.071254.

21. S. Tomovic, K. Yoshigoe, I. Maljevic, I. Radusinovic, Software-defined fog network architecture for IoT, *Wireless Pers. Commun.*, **92** (2017), 181–196. doi: 10.1007/s11277-016-3845-0.

22. Z. J. Qin, G. Denker, C. Giannelli, P. Bellavista, N. Venkatasubramanian, A software defined networking architecture for the internet-of-things, in *2014 IEEE Network Operations and Management Symposium (NOMS)*, (2014), 1–9. doi: 10.1109/NOMS.2014.6838365.

23. C. Gonzalez, O. Flauzac, F. Nolot, A. Jara, A novel distributed SDN-secured architecture for the IoT, in *2016 International Conference on Distributed Computing in Sensor Systems (DCOSS)*, (2016), 244–249. doi: 10.1109/DCOSS.2016.22.

24. M. Nobakht, V. Sivaraman, R. Boreli, A host-based intrusion detection and mitigation framework for smart home IoT using OpenFlow, in *2016 11th International Conference on Availability, Reliability and Security (ARES)*, (2016), 147–156. doi: 10.1109/ARES.2016.64.

25. O. Salman, S. Abdallah, H. I. Elhajj, A. Chehab, A. Kayssi, Identity-based authentication scheme for the internet of things, in *2016 IEEE Symposium on Computers and Communication (ISCC)*, (2016), 1109–1111. doi: 10.1109/ISCC.2016.7543884.

26. T. H. Nguyen, M. Yoo, A hybrid prevention method for eavesdropping attack by link spoofing in software-defined Internet of Things controllers, *Int. J. Distrib. Sensor Networks*, **13** (2017), 1550147717739157. doi: 10.1177/1550147717739157.

27. P. Bull, R. Austin, E. Popov, M. Sharma, R. Watson, Flow based security for IoT devices using an SDN gateway, in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, (2016), 157–163. doi: 10.1109/FiCloud.2016.30.

28. M. E. Ahmed, H. Kim, DDoS attack mitigation in Internet of Things using software defined networking, in *2017 IEEE Third International Conference on Big Data Computing Service and Applications (BigDataService)*, **2017** (2017), 271–276. doi: 10.1109/BigDataService.2017.41.

29. F. De Rango, G. Potrino, M. Tropea, P. Fazio, Energy-aware dynamic internet of things security system based on elliptic curve cryptography and message queue telemetry transport protocol for mitigating replay attacks, *Pervasive Mobile Comput.*, **67** (2020), 101105. doi: 10.1016/j.pmcj.2019.101105

30. F. De Rango, M. Tropea, P. Fazio, Mitigating DoS attacks in IoT EDGE Layer to preserve QoS topics and nodes' energy, in *IEEE INFOCOM 2020—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, (2020), 842–847. doi: 10.1109/INFOCOMWKSHPS50562.2020.9162902.

31. F. Song, L. Li, I. You, H. Zhang, Enabling geterogeneous deterministic networks with smart collaborative theory, *IEEE Network*, **35** (2021), 64–71. doi: 10.1109/MNET.011.2000613.

32. F. Song, Z. Ai, Y. Zhou, I. You, R. Choo, H. Zhang, Smart collaborative automation for receive buffer control in multipath industrial networks, *IEEE Trans. Ind. Inf.*, **16** (2020), 1385–1394. doi: 10.1109/TII.2019.2950109.

33. A. Hakiri, P. Berthou, A. Gokhale, S. Ellatif, Publish/subscribe-enabled software defined networking for efficient and scalable IoT communications, *Commun. Mag. IEEE*, **53** (2015), 48–54. doi: 10.1109/MCOM.2015.7263372.