



## Research Article

# A SYN Flood Attack Detection Method Based on Hierarchical Multihead Self-Attention Mechanism

Xiaojun Guo <sup>1,2,3</sup> and Xuan Gao <sup>1,2,3</sup>

<sup>1</sup>School of Information Engineering, Xizang Minzu University, Xianyang 712082, China

<sup>2</sup>Key Laboratory of Optical Information Processing and Visualization Technology of Tibet Autonomous Region, Xianyang 712082, China

<sup>3</sup>Xizang Cyberspace Governance Research Center, Xianyang 712082, China

Correspondence should be addressed to Xiaojun Guo; aikt@xzmu.edu.cn

Received 21 November 2021; Accepted 25 August 2022; Published 14 September 2022

Academic Editor: Shah Nazir

Copyright © 2022 Xiaojun Guo and Xuan Gao. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Existing SYN flood attack detection methods have obvious problems such as poor feature selectivity, weak generalization ability, easy overfitting, and low accuracy during training. In the paper, we present a SYN flood attack detection method based on the Hierarchical Multihead Self-Attention (HMHSA) mechanism. First, we use one-hot encoding and normalization to preprocess traffic data. Then the preprocessed traffic data is transmitted to the Feature-based Multihead Self-Attention (FBMHA) layer for feature selection. Finally, we use data slices to determine the features of the preprocessed traffic data under time series by passing the preprocessed traffic data into the Slice-based Multihead Self-Attention (SBMHA) layer. We tested the proposed method on different datasets. The experimental results show that compared with other works, our method presents better in feature selection and higher detection accuracy (even up to 99.97%).

## 1. Introduction

With the development of the shared and open Internet, network security is facing unprecedented challenges. Distributed denial of service (DDoS) attack has been a challenge for cyberspace security, and its number, frequency, complexity, and impact of DDoS are proliferating. The attack methods become particularly difficult to mitigate [1]. SYN flood attack is one of the most popular DDoS attack methods mainly exploiting the three-way handshake defect in the TCP protocol and IP spoofing techniques.

The three-way handshake mechanism establishes the TCP connection between the client and the server. In order to establish a TCP connection, the client must send a synchronize (SYN) packet to the server. After receiving the SYN message sent from the client, the server returns a SYN-ACK packet. When the client receives the SYN-ACK packet, it sends an ACK packet to the server. So far, the three-way handshake is completed [2].

An attacker exploits the server's half-opened connection state (SYN\_RECV) to perform the SYN flood attack on the server. The attacker sends a large number of SYN request packets with forged source IP addresses. The server treats these requests as legitimate. First, the server allocates memory and resources for these IP sources. Then, it sends the SYN-ACK packet to the client and finally waits for the client's ACK packet in the half-opened state. Attackers send a large number of illegal SYN requests to cause the TCP backlog queue to overflow and create half-opened connections until the system resources are exhausted. Many operating systems and even firewalls and routers are unable to defend against this attack effectively, and SYN flood attacks have a huge impact on fields such as finance, education, and media. The attack principle can be depicted as in Figure 1.

Deep learning provides a new idea for the study of network anomaly traffic detection. However, the information stored by encoded vectors and their relationship is

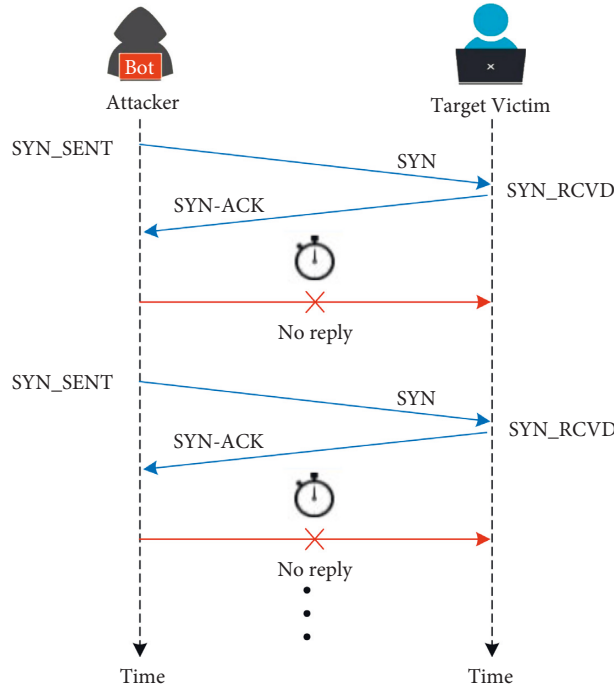


FIGURE 1: SYN flood attack.

limited by the distance between sequences for long-term sequences, and it results in the loss of important features between sequences.

Therefore, we propose an SYN flood attack detection method based on the HMHSA mechanism. The method uses Bidirectional Gated Recurrent Unit (Bi-GRU) neural network to encode the input sequence and fully considers the influence before and after information of each attribute. Then we add the Multihead Self-Attention mechanism to learn dependencies between sequences and extract salient features. In the self-attention mechanism, each datum is needed to be calculated with attention with all data. No matter how long the distance is, the maximum path length is also 1. Therefore, it can better capture long-distance dependence. Moreover, the multihead can learn relevant information from different representation subspaces to improve feature selection. The experimental results verify the effectiveness of the HMHSA mechanism. Compared with other methods, it improves the accuracy of SYN flood attack detection.

The main contributions of this paper include those as follows:

- (1) We apply double-layer Bi-GRU in encoding input sequence, which has fewer parameters, good feature selectivity, and can improve to weak adaptability of network
- (2) We add the Multihead Self-Attention mechanism to highlight important features. Compared with the attention mechanism, the Multihead Self-Attention can learn the deep feature information of a long sequence and can improve the accuracy while preventing over-fitting

- (3) We verify the generalization ability of our method with three different datasets, the CICDDoS2019 dataset [29], Mirai dataset [3], and the TDS\_SELF dataset (self-made traffic dataset), and the accuracy was up to 99.97%

The rest of this paper is organized as follows. In Section 2, we discuss relevant work in the domain of SYN flood attack detection. Section 3 details the proposed methodology. The results and analysis of experiments are given in Section 4. Finally, Section 5 describes the conclusion of this paper and the direction of future work.

## 2. Related Work

At present, the research on SYN flood attack detection can be divided into three categories: statistical methods [4, 5], machine learning methods [6–9], and deep learning methods. The statistical methods require feature vector extraction based on professional knowledge, but the uncertainty of human factors may affect accuracy. Due to traditional machine learning limitations, it is impossible to obtain the deep features from the long-term sequence of attack traffic.

In recent years, deep learning has been effective in DDoS detection and SYN flood detection. In 2017, Yuan et al. [10] proposed a DDoS detection method based on the long short-term memory (LSTM) network using the UNB ISCX 2012 dataset. It detects abnormal traffic by extracting 20 fields from a sequence of continuous-flow packets and using a sliding time window. Brun et al. [11] proposed a deep learning method based on the dense random neural network, analyzed the SYN flood attack on the IoT network, determined the

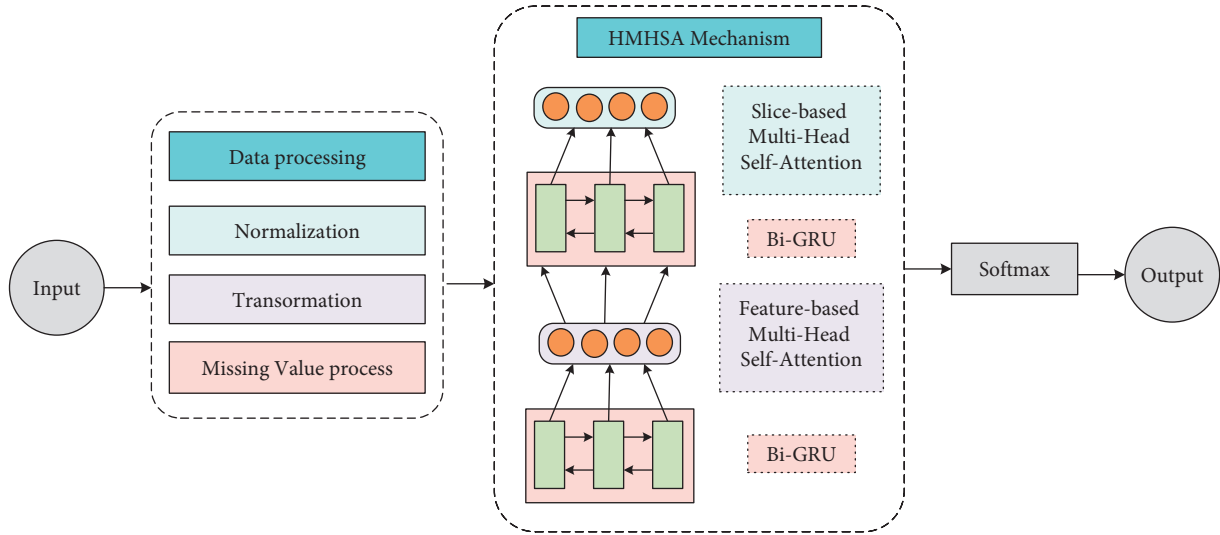


FIGURE 2: Architecture of the HMHSA mechanism.

related indicators of different attacks, and explained how to calculate features from packet capture. Li and Lu [12] developed a DDoS detection mechanism based on LSTM and Bayes (LSTM-BA). Through the LSTM method, partial DDoS attacks with high confidence output were identifiable. For those outputs with low confidence, the Bayes method was further used for secondary judgment to improve accuracy. Shaahan et al. [13] proposed to apply a convolutional neural network (CNN) for DDoS detection. The results show that CNN achieves superior performance compared to traditional machine learning methods. However, the CNN convolution kernel still needed to be optimized. Asad et al. [14] used feedforward backpropagation architecture and seven hidden layers to classify network flows. Evmorfos et al. [15] compared the random neural network with the LSTM. The experimental results show that the random network provides better attack detection and significantly reduces error rate [10]. Odumuyiwa et al. [16] compared the effect of unsupervised learning algorithms in DDoS detection and found that autoencoder works the best, but it needed to be replicated in larger systems to detect damaged endpoints. Nagaraju et al. [17] proposed a binary fruit fly algorithm for the real-time prediction model of SYN flood attack, which used swarm intelligence to find the optimal parameters. However, the authors only trained on one dataset, which cannot verify the generalization ability of the proposed model. Rehman et al. [18] proposed and evaluated four anomaly detection algorithms for DDoS attacks, which were gated recurrent unit (GRU), recurrent neural networks (RNN), naïve Bayes (NB), and sequential minimal optimization (SMO). SMO had the best effect on SYN flood detection. Britto and Priya [19] proposed an improved DDoS attack detection method in the cloud. It uses the deep belief network and the support vector machine (SVM) as a learning mechanism to improve detection accuracy. The XGBoost had been used as a classification model in the literature [20, 21]. The XGBoost algorithm has higher accuracy and lower false positive rate than other algorithms. In

addition, XGBoost detects extraordinarily quickly. Ravi et al. [22] proposed AEGIS (a similarity measure) to detect and mitigate SYN flood attack for SDN controllers through regular checks. Besides, Wan et al. [23] proposed a similarity measure. According to the similarity of the data collected by the nodes, the correlation function is defined in the fuzzy theory to classify nodes and select redundant nodes. The PSO method [24–27] was also used to detect SYN flood attacks. This defense strategy improves the performance of the system in both memory usage and attack request dwell time.

Through the above related research, it can be seen that the LSTM method is used the most, but the LSTM parameter is too large and the training speed is slow. The accuracy of other methods needs to be improved. Besides, most of these studies do not consider the impact of time series on SYN flood attack detection.

### 3. Proposed Methodology

First, the HMHSA mechanism performs data preprocessing, including missing value processing, data transformation, and normalization. Then, we build and train the HMHSA mechanism. Finally, the vector with high weight is extracted, and the classification results are obtained by the Softmax. The overall architecture of the HMHSA mechanism is shown in Figure 2.

**3.1. Bi-GRU Neural Network.** GRU is a lightweight version of LSTM. GRU has only two gate structures, namely, update gate and reset gate. The structure of the GRU neural unit is depicted as in Figure 3.

The GRU uses the update gate to store the amount of information saved from the previous memory to the current timestep and uses the reset gate to determine how to combine the new input information with the historical information [28]. A GRU network is defined as

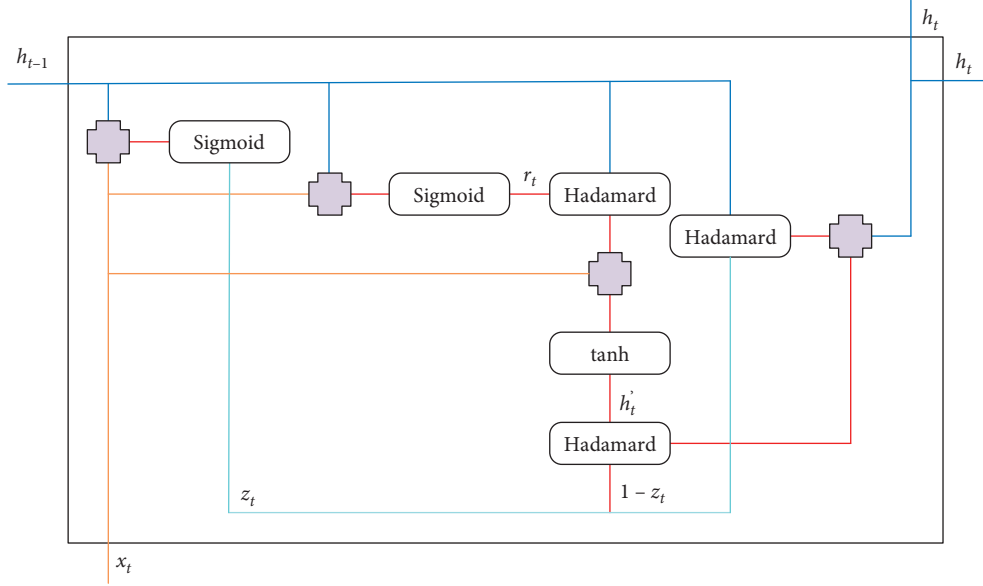


FIGURE 3: Structure of GRU.

$$\begin{aligned}
 z_t &= \text{Sigmoid}(W^{(z)}x_t + U^{(z)}h_{t-1}), \\
 r_t &= \text{Sigmoid}(W^{(r)}x_t + U^{(r)}h_{t-1}), \\
 h'_t &= \tanh(Wx_t + r_t \odot U h_{t-1}), \\
 h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot h'_t,
 \end{aligned} \tag{1}$$

where  $z_t$  is the activation result of the update gate. The input vector  $x_t$  of the  $t$  timestep and the information  $h_{t-1}$  of the previous timestep are linearly transformed and then are put into the Sigmoid activation function, which maps the variables between 0 and 1.  $r_t$  is the result of reset gate, which measures the opening size of the gate.  $h'_t$  calculates the current memory content by reset gate and uses the Hadamard function to determine the previous information to be retained and forgotten, where  $\odot$  represents the Hadamard product.  $h_t$  is the final updated node status.

The advantage of GRU is that it can discard and retain the information in the dimension simultaneously by using a gate  $z_t$ . In this paper, we used a bidirectional GRU to process the input sequence forward and backward in turn so that the output node of each timestep contains the complete past and future information under the current moment in the input sequence. The Bi-GRU is given by

$$\begin{aligned}
 \vec{h}_t &= \overrightarrow{\text{GRU}}(x_t), t \in [1, T], \\
 \overleftarrow{h}_t &= \overleftarrow{\text{GRU}}(x_t), t \in [1, T], \\
 h_t &= \left[ \vec{h}_t, \overleftarrow{h}_t \right].
 \end{aligned} \tag{2}$$

**3.2. Multihead Self-Attention Mechanism.** Attention mechanisms can selectively focus on important information of the research subject. Using the transformer model for reference, we use the Multihead Self-Attention mechanism to extract

the dependencies in the sequence space. This mechanism captures the information of the same sequence in different subspaces by combining multiple parallel self-attention calculations and then obtains more comprehensive correlation features from multiple perspectives and levels. The structure of the Multihead Self-Attention mechanism is shown in Figure 4, which mainly includes two parts.

- (1) Scaled dot-Product attention: The scaling weight prevents the vector dimension from being too high to cause the calculated dot product result to be too large as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{3}$$

The inputs are composed of query ( $Q$ ), key ( $K$ ), and value ( $V$ ) matrix. When  $Q = K = V$ , it is self-attention mechanism.  $QK^T$  is the attention matrix, and  $\sqrt{d_k}$  turns the attention matrix into a standard normal distribution.

- (2) Attention calculation: The original  $Q$ ,  $K$ , and  $V$  are linear mapped several times, and the result of each mapping is input into the scaled dot-product attention, and the result obtained each time is called a head, which is computed as

$$\begin{aligned}
 \text{head}_i &= \text{Attention}(Q \cdot W_i^Q, K \cdot W_i^K, V \cdot W_i^V), \\
 \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \text{head}_2, \dots, \text{head}_n).
 \end{aligned} \tag{4}$$

**3.3. The HMHSA Mechanism.** The HMHSA mechanism consists of two layers—Bi-GRU, and each layer introduces a Multihead Self-Attention mechanism. Feature-based Multihead Self-Attention is used to enhance the expression of

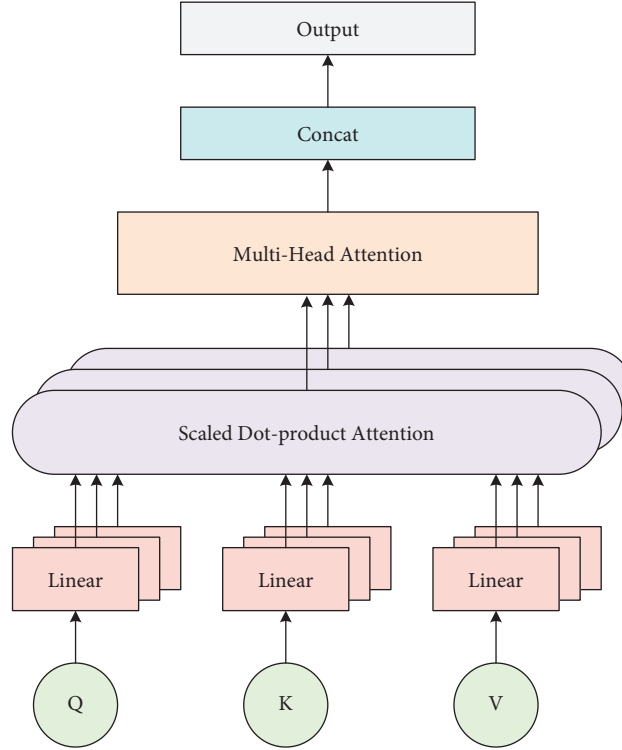


FIGURE 4: Structure of multihead self-attention.

traffic features, and Slice-based Multihead Self-Attention is used for grouping traffic data. Figure 5 shows the structural outline of the HMHSA mechanism.

**3.3.1. FBMHA Layer.** Not all features are equally important, so to fully capture the significant features, we used this mechanism to determine which features should be the focus.

- (1) First, an  $N$ -dimensional sample is given:  $X_i = [x_i^0, x_i^1, \dots, x_i^{N-1}]$ , next the word vector of byte data is encoded by Bi-GRU neural network, then the characteristics of byte data are learned and finally the  $h_i^j$  is generated:

$$\begin{aligned} \vec{h}_i^j &= \overrightarrow{\text{GRU}}(x_i), \\ \overleftarrow{h}_i^j &= \overleftarrow{\text{GRU}}(x_i), \\ h_i^j &= \left[ \vec{h}_i^j, \overleftarrow{h}_i^j \right]. \end{aligned} \quad (5)$$

- (2) The FBMHA mechanism is introduced to calculate the weight distribution of sequence data, and the sequence information with important contributions is highlighted. The input vector comes from the output vector of the Bi-GRU layer;  $d$  is the dimension of  $Q$  and  $K$  vector; linear transformation of  $Q$ ,  $K$ , and  $V$  with different parameters  $W_i^Q$ ,  $W_i^K$ , and  $W_i^V$ . We set the number of heads  $h$  as 2, after two calculations of scaled dot product attention, the output weight matrix  $W^o$  is used to connect two parallel heads to obtain result  $a_i^j$ :

$$Q, K, V = h_i^j,$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK}{\sqrt{d}}\right)V,$$

$$b_h^i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V),$$

$$a_i^j = \text{concat}(b_h^1, b_h^2, \dots, b_h^N)W^o. \quad (6)$$

After obtaining the weight matrix of the sequence data, the weighted sum of the weight matrix  $a_i^j$  and the byte information feature vector  $h_i^j$  are calculated, and the feature representation of each byte is updated to obtain the final output  $S_i$ :

$$S_i = \sum a_i^j h_i^j. \quad (7)$$

**3.3.2. SBMHA Layer.** Intrusion detection traffic data are related to time. The traffic information of multiple adjacent matrices helps to determine the type of current traffic. Traffic data is grouped and called data slicing. When each traffic group is synthesized into a large data packet, the Bi-GRU network ignores the important influence of some key data packet information on the classification results. By introducing the attention mechanism, the weight distribution of the data packets is calculated, and the traffic information with important contributions is highlighted in a group.

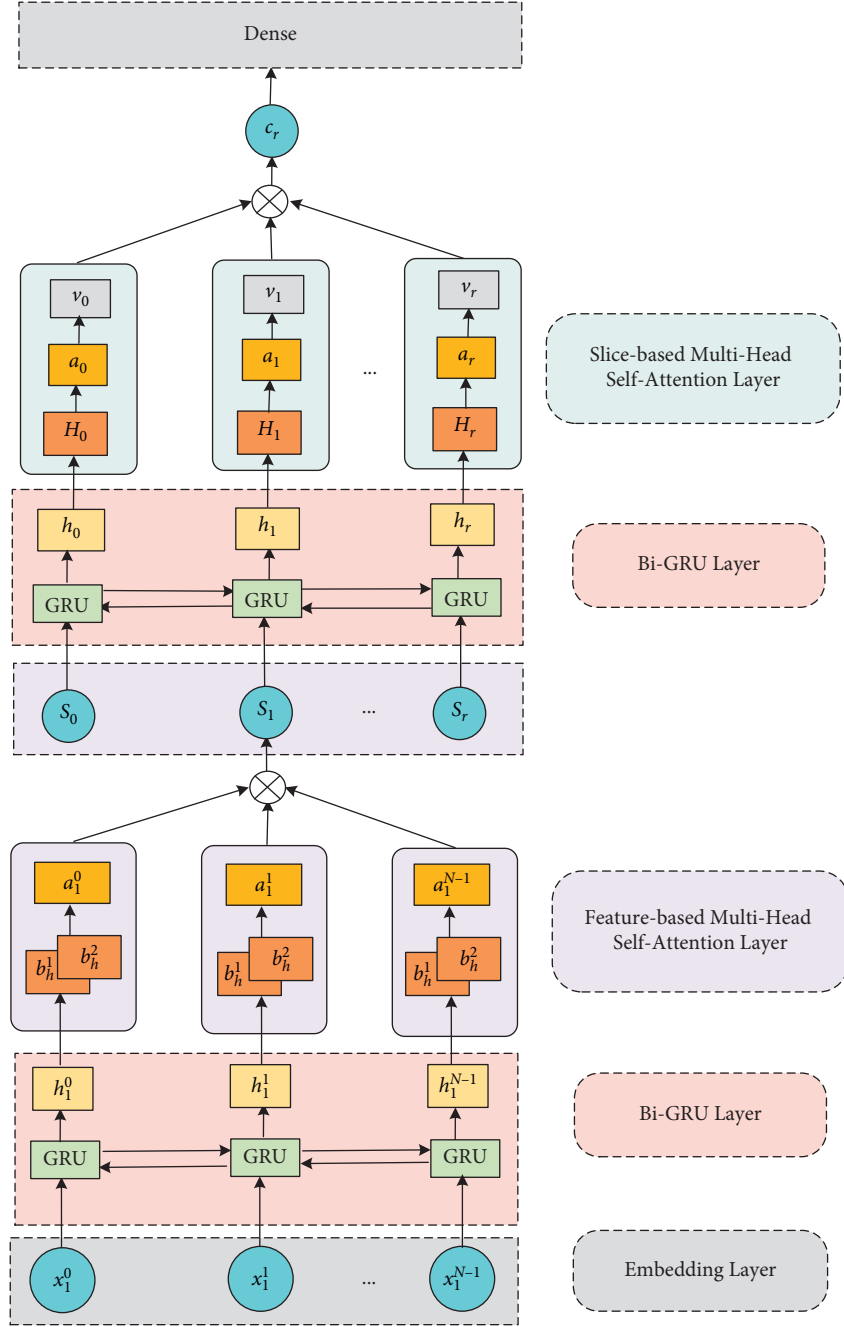


FIGURE 5: HMHSA mechanism.

- (1) The  $S_i$  generated by the upper layer is used to generate the characteristic vector  $h_i$  of the network flow through Bi-GRU neural network:

$$\begin{aligned}
 \vec{h}_i^j &= \overrightarrow{\text{GRU}}(S_i), \\
 \overleftarrow{h}_i^j &= \overleftarrow{\text{GRU}}(S_i), \\
 h_i &= \begin{bmatrix} \vec{h}_i^j \\ \overleftarrow{h}_i^j \end{bmatrix}.
 \end{aligned} \tag{8}$$

- (2) Introducing a SBMHA mechanism: For each time-step, the corresponding hidden state  $h_i$  is fed through a single layer of perception to obtain  $u_i$  as the hidden representation of  $h_i$ . Similarly, through the SBMHA mechanism, the similarity is used to evaluate the importance of each flow at different times. Finally, the weighted sum is calculated, which is expressed as the context vector  $v_i$ :

$$u_i = \tanh(W_w h_i + b_w), \tag{9}$$

where  $W_w$  and  $b_w$  represent the weight vector and the bias term, respectively.

$$H_i = \text{softmax}(u_i^T u_s) = \frac{\exp(u_i^T u_s)}{\sum_i \exp(u_i^T u_s)}. \quad (10)$$

We evaluate the importance of each slice at different times using the similarity of  $u_i$  and  $u_s$ , where  $u_s$  is the adjacent slice traffic vector.

$$a_i = \text{concat}(H_1, H_2, \dots, H_i). \quad (11)$$

The fusion feature  $a_i$  is obtained through the Multihead Self-Attention layer.

$$v_i = \sum_i a_i h_i. \quad (12)$$

where  $v_i$  is the weighted sum of the weight matrix  $a_i$  and the data flow information feature vector  $h_i$ .

A summary of the algorithmic phases of the HMHSA mechanism in Algorithm 1 is provided below.

## 4. Experiment

**4.1. Dataset.** In order to verify the performance of HMHSA mechanism, experiments are carried out on three datasets, and the data statistics are shown in Table 1. The CICDDoS2019 dataset [29], Mirai dataset [3], and the simulated real dataset (self-made traffic dataset, TDS\_SELF) are used, respectively.

**4.1.1. CICDDoS2019 Dataset.** The CICDDoS2019 dataset contains normal traffic and the latest common DDoS attacks. At present, many DDoS attack detections [30–34] are based on this dataset. The results of the network traffic analysis use CICFlowMeter-V3, which contains traffic based on timestamps, source IP and destination IP, source port and destination port, protocols, attack types, and other markers and extracted more than 80 traffic features [29]. By analyzing the SYN flood traffic characteristics, Table 2 shows that the characteristics selected from the CICDDoS2019 dataset are suitable for our experiment.

**4.1.2. Mirai Dataset.** The Mirai dataset is created by Meidan et al. [3]; Mirai is a specific type of botnet malware that overrides networked Linux devices and successfully turns them into bots used for distributed attacks such as DDOS. The Mirai dataset contains a large number of SYN flood instances.

**4.1.3. TDS\_SELF Dataset.** TDS\_SELF dataset is constituted of our real local network traffic and SYN flood attack traffic. The SYN flood attack traffic is generated through “hping3” simulation, as shown in Figure 6. Wireshark is used to capture packets, and tcp.srcport, tcp.dstport, tcp.flags.ack, tcp.flags.syn, tcp.flags.fin, etc. are selected as features through CICFlowMeter, as shown in Figure 7.

**4.1.4. Data Preprocessing.** Preprocessing mainly includes feature transformation and feature normalization. Feature conversion uses one-hot encoding to digitize the sequence, and one-hot encoding can extend the value of discrete features to Euclidean space, making the distance calculation between features more reasonable [35]. We use the Min-Max method for normalization, where min is the minimum value of the sample data and max is the maximum value of the sample data. The Min-Max formula is as follows:

$$x^* = \frac{x - \min}{\max - \min}. \quad (13)$$

**4.2. Evaluation.** In order to evaluate the performance of the HMHSA mechanism in the detection of SYN flood attack, we use four indicators: accuracy, precision, recall, and F1-score [36].

- (1) *Accuracy.* Measure the proportion of the model’s correct prediction of the samples in the dataset:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}. \quad (14)$$

- (2) *Precision.* Measure the proportion of the sample that is predicted to be positive:

$$\text{Precision} = \frac{TP}{TP + FN}. \quad (15)$$

- (3) *Recall.* Evaluate whether to find all the true positive examples in the sample:

$$\text{Recall} = \frac{TP}{TP + FP}. \quad (16)$$

- (4) *F1-score.* The balance between accuracy and recall:

$$F1 - \text{score} = 2 \left( \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \right). \quad (17)$$

**4.3. Experimental Configuration.** The method proposed in this paper is verified and implemented on PC, and the experimental environment is Lenovo Legion R7000; CPU R7-4800H; memory 16G, hard disk 512G; operating system Windows 19H2; compile PyCharm 2021, Python 3.6; and the neural network framework is Tensorflow 1.8.0, Keras 2.1.6.

For the parameter setting, we obtained the optimal parameters through several experiments, and the basic parameter settings of the method are shown in Table 3.

**4.4. Experimental Analysis.** In this paper, four experiments were conducted: (1) the determination of the number of heads, and the suitable number of heads were selected by comparing the F1-score; (2) when the epoch is 4, it tended to be stable, indicating that the method had a good effect, and small number of training times can achieve high accuracy; (3) the timestep was selected randomly for the data slicing layer to discuss the influence on the convergence performance; (4) the comparison of different attentions further verified the superiority of choosing attention in this study.

```

(1) Start
(2) Input: Training dataset  $X_i = [x_i^0, \dots, x_i^{N-1}]$ , epochs  $K$ , timesteps  $N_t$ 
(3) Output: The classification category  $y$ 
(4) Data preprocessing: Missing value filling, numerical conversion, normalization
(5) For  $k=1: K$  do
(6) //Learning byte data feature information through Bi-GRU
(7) Calculate  $h_i^j$ 
(8) //Byte data weight distribution obtained by Multihead Self-Attention mechanism
(9)  $Q, K, V \leftarrow h_i^j$ 
(10) Calculate Attention ( $Q, K, V$ ),  $b_h^i, a_i^j$ 
(11) //Update each byte feature representation to get the data flow vector  $S_i$ 
(12)  $S_i \leftarrow \sum a_i^j h_i^j$ 
(13) The current data flow  $S_i$  is merged with the history data  $S_t$ , where the length of history data is determined by  $N_t$ 
(14) //Learning data flow feature information through Bi-GRU
(15) Calculate  $h_i$ 
(16) //Calculation of data flow weight distribution by Multihead Self-Attention
(17) Calculate  $u_i, H_i, a_i$ 
(18) //Calculate the weighted sum
(19)  $v_i \leftarrow \sum a_i h_i$ 
(20) Train the model
(21) The output  $o_t$  of the model is obtained
(22) if  $o_t > 0.5$  then
(23)    $y_t = 1$  //SYN Flood attack
(24) else
(25)    $y_t = 0$  //benign data
(26) end if
(27) end for
(28) return  $y_t$ 
(29) END

```

ALGORITHM 1: The HMHSA mechanism.

TABLE 1: Composition of dataset.

		CICDDoS2019	Mirai	TDS_SELF
Train samples	Normal samples	37597	56850	40000
	Anomaly samples	45332	56850	40000
	Total samples	82930	113700	80000
Test samples	Normal samples	45714	54548	50000
	Anomaly samples	109023	54548	60000
	Total samples	154737	109096	110000

TABLE 2: CICDDoS2019 dataset.

Features	Describe
Source/destination port	Source/destination port
Flow bytes/s	The number of packet bytes transmitted per second
Flow packets/s	The number of packets transmitted per second
Flow IAT mean	Average rate
Fwd PSH flags	The number of times the PSH flag is set in a forward transmitted packet
Bwd PSH flags	The number of times the PSH flag is set in a packet transmitted in reverse
Fwd URG flags	The number of times the URG flag is set in a forward transmitted packet
Bwd URG flags	The number of times the URG flag is set in reverse packet
FIN flag count	The number of packages with FIN
SYN flag count	The number of packages with SYN
RST flag count	The number of packets with RST
PSH flag count	The number of packages with PUSH
ACK flag count	The number of packets with ACK
URG flag count	The number of packages with URG
act_data_pkt_fwd	Packets with a TCP data payload of at least 1 byte in the forward direction
Active mean	Average time a stream is active before it is idle
Active std	Standard deviation time for a stream to be active before it is idle



```
kali@kali:~/Desktop$ hping3
hping3> sudo hping3 -q --rand-source -S --keep --flood 192.168.0.103
HPING 192.168.0.103 (eth0 192.168.0.103): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

FIGURE 6: Hping3 simulates SYN flood attack.

Time	Source	Destination	Protocol	Length	Info
0.000000000	100.170.137.131	192.168.0.103	TCP	56	98 → 0 [SYN] Seq=0 Win=512 Len=0
1.001151630	32.185.168.113	192.168.0.103	TCP	56	99 → 0 [SYN] Seq=0 Win=512 Len=0
2.002307452	168.162.118.1	192.168.0.103	TCP	56	100 → 0 [SYN] Seq=0 Win=512 Len=0
3.003721272	188.253.15.188	192.168.0.103	TCP	56	101 → 0 [SYN] Seq=0 Win=512 Len=0
4.005430482	0.10.79.86	192.168.0.103	TCP	56	102 → 0 [SYN] Seq=0 Win=512 Len=0
5.006628472	4.143.7.75	192.168.0.103	TCP	56	103 → 0 [SYN] Seq=0 Win=512 Len=0
6.007152701	40.202.160.133	192.168.0.103	TCP	56	104 → 0 [SYN] Seq=0 Win=512 Len=0
7.011359338	36.148.94.215	192.168.0.103	TCP	56	105 → 0 [SYN] Seq=0 Win=512 Len=0
8.014850282	126.109.179.15	192.168.0.103	TCP	56	106 → 0 [SYN] Seq=0 Win=512 Len=0

FIGURE 7: SYN flood attack traffic captured by wireshark.

TABLE 3: Basic parameter settings of the method.

Parameters	Values
Batch_size	1024
Epoch	6
Dropout	0.5
Learning_rate	0.001
Timestep	3
Two-layer bi-GRU unit	32, 12
Number of attention heads	2
Optimizer	Adam

**4.4.1. Determination of the Number of Heads.** The number of hyperparameter heads affects the attention of different features. The appropriate number of heads can extract the key spatial characteristics of data packets more accurately. Too many or very few heads may cause the lack or interference of effective features. In this paper, we set the number of heads ( $N_{head}$ ) as 1–6, and the experimental results are shown in Figure 8. It can be seen that when the number of heads is 2, the classification result or  $F1$ -score is better than other [29] heads [3].

**4.4.2. Determination of Timestep.** By comparing the loss values of the three datasets at different timesteps, it is found that the minimum loss values are 0.0010, 0.0012, and 0.0026 when the timestep is 3. When the timestep exceeds 3, the loss will increase to some extent so that the timestep can be selected as 3 (see Figure9).

**4.4.3. Analysis of Training Results.** The HMHSA mechanism was trained on the CICDDoS2019 dataset and TDS\_SELF dataset. The results are shown in Figures 10(a) and 10(b), where  $Acc\_Train$  and  $Acc\_Test$  are the accuracies of the training set and the testing set, respectively. As can be seen from the figures, when epoch is 4, the highest accuracies are 99.96% and 99.97%, respectively, and the accuracies tend to be stable.

In order to further verify the generalization ability of the model, we trained on the Mirai dataset and added an Early Stopping function to prevent overfitting. If the training effect is still not improved after a certain number of times, the training is stopped. We adjusted the upper limit of epochs to 100 and stopped training in advance when loss did not decrease for seven consecutive epochs during network training, which further improved the network fit. The experimental result is shown in Figure 10(c). The accuracy of the model on the Mirai dataset can reach [29] 99.97%. [3].

**4.4.4. Comparison of Different Attention Mechanisms.** In order to evaluate the effectiveness of the HMHSA mechanism, we designed seven different attention mechanisms for comparison. They are No Attention (Bi-GRU), Single Attention (SA), Single Self-Attention (SSA), Single Multihead Self-Attention (SMHSA), Hierarchical Attention (HA), Hierarchical Self-Attention (HASA), and HMHSA mechanism. The experiments are carried out on three datasets. The accuracy, precision, recall, and  $F1$ -score of different structures are shown in Figures 11–13. Comparative experimental results show that the proposed method can achieve good results in SYN flood attack detection [29].

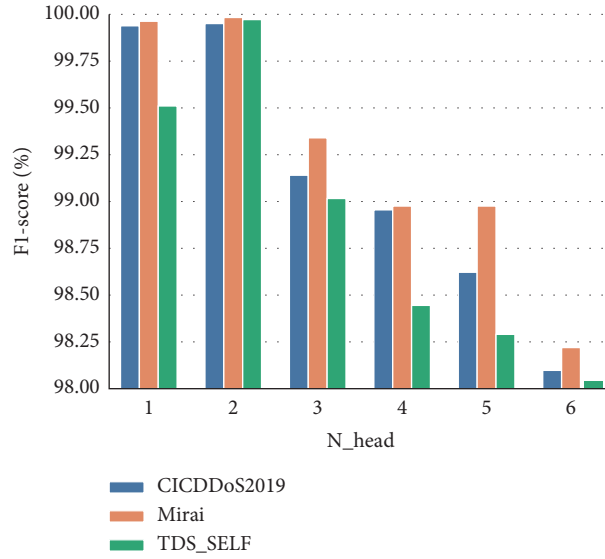


FIGURE 8: Comparison of F1-score with different number of heads on the CICDDoS2019, Mirai, and TDS\_SELF dataset.

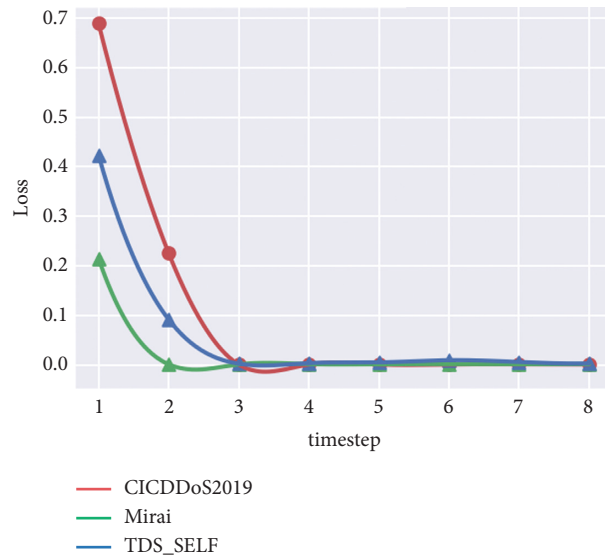


FIGURE 9: Comparison of loss at different timesteps on the CICDDoS2019 [29], Mirai [3], and TDS\_SELF dataset.

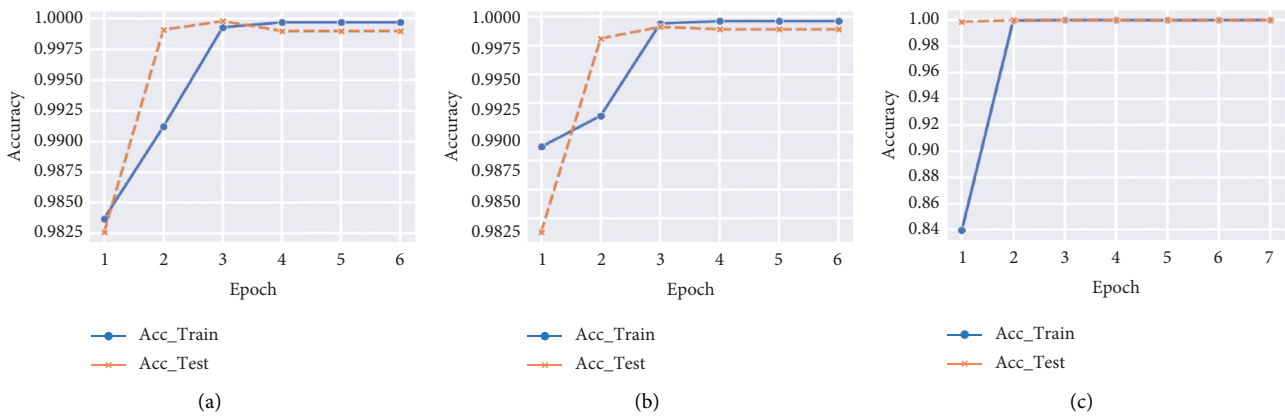


FIGURE 10: Performance test on the CICDDoS2019, Mirai, and TDS\_SELF dataset. (a) CICDDoS2019. (b) Mirai. (c) TDS\_SELF.

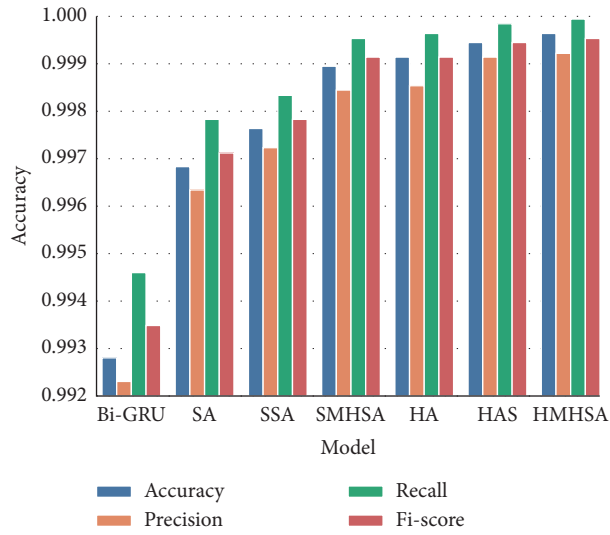


FIGURE 11: Comparison of different attention mechanisms on the CICDDoS2019 dataset.

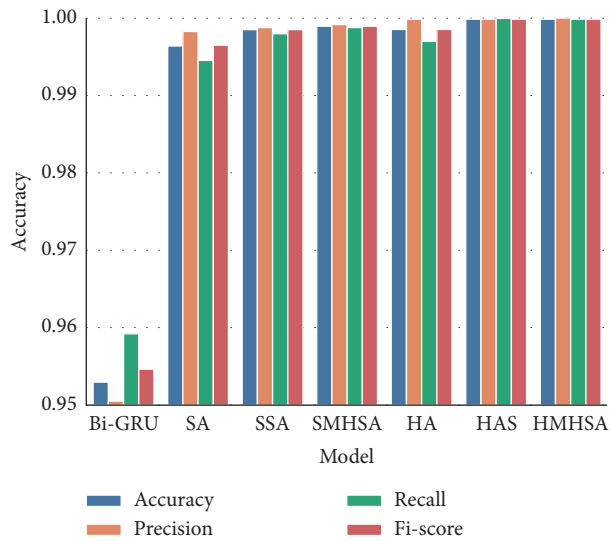


FIGURE 12: Comparison of different attention mechanisms on the Mirai dataset.

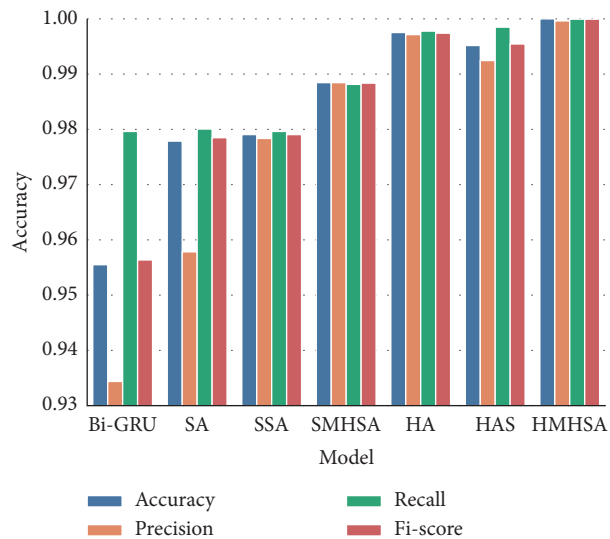


FIGURE 13: Comparison of different attention mechanisms on the TDS\_SELF dataset.

## 5. Conclusion

In this study, we have proposed a SYN flood attack detection method [3] with Hierarchical Multihead Attention mechanism. First, Bi-GRU is used to learn the feature information of byte data, and further, the weight distribution of byte data is calculated through the Multihead Self-Attention mechanism to capture the internal correlation of byte data and to highlight the important contribution byte information. Then, control the value of timestep to perform traffic slicing, merge historical data flow and current data flow, learn data flow feature information through Bi-GRU, and further calculate data flow weight distribution through Multihead Self-Attention. Finally, the classification is performed by the Softmax function.

Results illustrate that the method can perform better feature selection and improve the accuracy. Experiments are performed on the public network dataset CICDDoS2019, Mirai dataset, and the simulated TDS\_SELF dataset, and the accuracy can reach 99.96% (CICDDoS2019), 99.97% (Mirai dataset), and 99.97% (TDS\_SELF).

In future work, a fast and accurate defense mechanism is needed to detect TCP-SYN flood attacks, we plan to further optimize the model, reduce the network structure of the model, and design a more lightweight and more reliable model with a lower false positive rate. Due to the addition of the attention mechanism, our model lacks fast computation, and future goals will focus on balancing efficiency and accuracy to further study the evolution of attention. We plan to evaluate the performance of the proposed detection method against low-rate SYN flood attack when the attack is similar to the background total traffic. In addition to this, we will evaluate the proposed method on a real testbed using a larger-capacity real network traffic dataset.

## Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This work was supported by the Natural Science Foundation of Xizang Autonomous Region under Grant No. XZ2019ZRG-36(Z), the ZangQin Himalay Talent Development Support Program (Natural Science) for Distinguished Young Scholars under Grant No. 324011810216, the Research Team Project for Xizang-Related Network Information Content and Data Security under Grant No. 324042000709, and Xizang Cyberspace Governance Research Center.

## References

[1] A. Sangodoyin, B. Modu, I. Awan, and J. P. Disso, "An approach to detecting distributed denial of service attacks in

- software defined networks," in *Proceedings of the 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*, pp. 436–443, Barcelona, Spain, August 2018.
- [2] B. N. Subbulakshmi and T. Subbulakshmi, "Tcp syn flood attack detection and prevention system using adaptive thresholding method," *ITM Web of Conferences*, vol. 37, no. 4356, p. 01016, 2021.
- [3] Y. Meidan, M. Bohadana, Y. Mathov et al., "N-BaIoT-Network-Based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.
- [4] I. Ocovaj and S. Ocovaj, "Application of entropy formulas in detection of denial-of-service attacks," *International Journal of Communication Systems*, vol. 32, no. 15, p. e4067, 2019.
- [5] M. Rahouti, K. Xiong, N. Ghani, and F. Shaikh, "SYNGuard: dynamic threshold-based SYN flood attack detection and mitigation in software-defined networks," *IET Networks*, vol. 10, no. 2, pp. 76–87, 2021.
- [6] Y. Liu, M. Dong, K. Ota, J. Li, and J. Wu, "Deep reinforcement learning based smart mitigation of DDoS flooding in software-defined networks," in *Proceedings of the 2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pp. 1–6, IEEE, Barcelona, Spain, September 2018.
- [7] N. N. Tuan, P. H. Hung, N. D. Nghia, V. T. Nguyen, and V. P. Trung, "A robust TCP-SYN flood mitigation scheme using machine learning based on SDN," in *Proceedings of the 2019 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 363–368, IEEE, Jeju, Korea (South), October 2019.
- [8] R. Gegan, C. Mao, D. Ghosal, M. Bishop, and S. Peisert, "Anomaly detection for science DMZs using system performance data," in *Proceedings of the 2020 International Conference on Computing, Networking and Communications (ICNC)*, pp. 492–496, IEEE, Big Island, HI, USA, February 2020.
- [9] R. Swami, M. Dave, and V. Ranga, "Detection and analysis of TCP-SYN DDoS attack in software-defined networking," *Wireless Personal Communications*, vol. 118, no. 4, pp. 2295–2317, 2021.
- [10] X. Yuan, C. Li, and X. Li, "DeepDefense: identifying DDoS attack via deep learning," in *Proceedings of the 2017 IEEE International Conference on Smart Computing (SMART-COMP)*, pp. 1–8, IEEE, Hong Kong, China, May 2017.
- [11] O. Brun, Y. Yin, E. Kadioglu, G. J. Augusto, and M. Ramos, "Deep learning with dense random neural networks for detecting attacks against IoT-connected home environments," in *International ISCIS Security Workshop*, pp. 79–89, Springer, Cham, Berline Germany, 2018.
- [12] Y. Li and Y. Lu, "LSTM-BA: DDoS detection approach combining LSTM and Bayes," in *Proceedings of the 2019 Seventh International Conference on Advanced Cloud and Big Data (CBD)*, pp. 180–185, IEEE, Suzhou, China, September 2019.
- [13] A. R. Shaaban, E. Abd-Elwanis, and M. Hussein, "DDoS attack detection and classification via Convolutional Neural Network (CNN)," in *Proceedings of the 2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS)*, pp. 233–238, IEEE, Cairo, Egypt, December 2019.
- [14] M. Asad, M. Asim, T. Beg, H. MujtabaAbbas, and S. Abbas, "Deepdetect: detection of distributed denial of service attacks using deep learning," *The Computer Journal*, vol. 63, no. 7, pp. 983–994, 2020.

- [15] S. Evmorfos, G. Vlachodimitropoulos, N. Bakalos, and E. Gelenbe, "Neural network architectures for the detection of SYN flood attacks in IoT systems," in *Proceedings of the 13th ACM International Conference on Pervasive Technologies Related to Assistive Environments*, pp. 1–4, Corfu, Greece, June 2020.
- [16] V. Odumuyiwa and R. Alabi, "DDoS detection on Internet of things using unsupervised algorithms," *Journal of Cyber Security and Mobility*, vol. 10, pp. 569–592, 2021.
- [17] V. Nagaraju, A. Raaza, V. Rajendran, and D. Ravikumar, "Deep learning binary fruit fly algorithm for identifying SYN flood attack from TCP/IP," *Materials Today Proceedings*, vol. 65, 2021.
- [18] S. Rehman, M. Khaliq, S. I. Rasool, Z. ShafiqJavedJalilBashir, and A. K. Bashir, "DIDDOS: an approach for detection and identification of distributed denial of service (DDoS) cyber-attacks using gated recurrent units (GRU)," *Future Generation Computer Systems*, vol. 118, pp. 453–466, 2021.
- [19] J. D. Britto and M. S. Priya, "Deep belief network and support vector machine fusion for distributed denial of service and economical denial of service attack detection in cloud," *Concurrency and Computation: Practice and Experience*, vol. 34, p. e6543, 2021.
- [20] Y. Jiang, G. Tong, H. Yin, and N. Xiong, "A pedestrian detection method based on genetic algorithm for optimize XGBoost training parameters," *IEEE Access*, vol. 7, 2019.
- [21] Z. Chen, F. Jiang, Y. Cheng, G. Xin, and L. Weirong, "XGBoost classifier for DDoS attack detection and analysis in SDN-based cloud," in *Proceedings of the 2018 IEEE International Conference on Big Data and Smart Computing (Bigcomp)*, January 2018.
- [22] N. Ravi, S. M. Shalinie, C. Lal, and M. Conti, "AEGIS: detection and mitigation of TCP SYN flood on SDN controller," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 745–759, 2021.
- [23] R. Wan, N. Xiong, and N. T. Loc, "An energy-efficient sleep scheduling mechanism with similarity measure for wireless sensor networks," *Human-centric Computing and Information Sciences*, vol. 8, no. 1, p. 18, 2018.
- [24] Z. Ahmed, M. Mahbub, and S. Jahan, "Defense against SYN flood attack using LPTR-PSO: a three phased scheduling approach," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 9, pp. 8-9, 2017.
- [25] W. Guo, N. Xiong, H. C. Chao, S. Chen, and G. Chen, "Design and analysis of self-adapted task scheduling strategies in wireless sensor networks," *Sensors*, vol. 11, no. 7, pp. 6533–6554, 2011.
- [26] X. Wang, Q. Li, N. Xiong, and Y. Pan, "Ant colony optimization-based location-aware routing for wireless sensor networks," *International Conference on Wireless Algorithms, Systems, and Applications*, Springer, Berlin, Heidelberg, 2008.
- [27] F. XiaXia, R. Hao, and Y. LiXiongYangZhang, "Adaptive GTS allocation in IEEE 802.15.4 for real-time wireless sensor networks," *Journal of Systems Architecture*, vol. 59, no. 10, pp. 1231–1242, 2013.
- [28] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," in *n*, <https://arxiv.org/abs/1409.2329>, 2014.
- [29] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *Proceedings of the 2019 International Carnahan Conference on Security Technology (ICCST)*, pp. 1–8, IEEE, Chennai, India, October 2019.
- [30] Y. Jia, F. Zhong, A. Alrawais, B. Gong, and X. Cheng, "Flowguard: an intelligent edge defense mechanism against IoT DDoS attacks," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9552–9562, 2020.
- [31] S. Sindian and S. Sindian, "An enhanced deep autoencoder-based approach for DDoS attack detection," *WSEAS Transactions on Systems and Control*, vol. 15, pp. 716–724, 2020.
- [32] H. Kousar, M. M. Mulla, P. Shettar, and D. G. Narayan, "Detection of DDoS attacks in software defined network using decision tree," in *Proceedings of the 2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT)*, pp. 783–788, IEEE, Bhopal, India, June 2021.
- [33] M. A. Salahuddin, M. F. Bari, H. A. Alameddine, V. Pourahmadi, and R. Boutaba, "Time-based anomaly detection using autoencoder," in *Proceedings of the 16th International Conference on Network and Service Management (CNSM)*, pp. 1–9, Izmir, Turkey, November 2020.
- [34] N. M. Yungaicela-Naula, C. Vargas-Rosales, and J. A. Perez-Diaz, "SDN-based architecture for transport and application layer DDoS attack detection by using machine and deep learning," *IEEE Access*, vol. 9, pp. 108495–108512, 2021.
- [35] K. Jiang, W. Wang, A. Wang, and H. Wu, "Network intrusion detection combined hybrid sampling with deep hierarchical network," *IEEE Access*, vol. 8, pp. 32464–32476, 2020.
- [36] Y. Lu, S. Wu, Z. Fang, N. Xiong, S. Yoon, and D. S. Park, "Exploring finger vein based personal authentication for secure IoT," *Future Generation Computer Systems*, vol. 77, pp. 149–160, 2017.