LSH-XGBoost based Network Congestion Detection Method for SSDN

Kaiyuan Tian¹, a, Jian Wang¹, b, Yuanyuan Liao¹, c, Dengke Xu², d, Baigen Cai¹, e

¹School of Electronics and Information Engineering, Signal &Communication Research Institute.
²Beijing Jiaotong University, China Academy of Railway Sciences Corporation Limited. Beijing, China.

a17120273@bjtu.edu.cn, bwangj@bjtu.edu.cn, c17120238@bjtu.edu.cn,
dxdk Jason@139.com, ebgcai@bjtu.edu.cn

Abstract. Signal Safety Data Network (SSDN) is essential to maintain the reliable transmission between station and station equipment, station and central signal equipment. Threats in SSDN include but not limited to poor performance in telecommunication and denial of service when network congestion occurs. Therefore, it’s of great significance detecting the potential network congestion in railroad SSDN. Existing detection algorithms are low efficient, less accurate and are unable to cover all the detection categories. By combining the advantages of low time complexity and high classification accuracy originate from Local Sensitive Hashing (LSH) and eXtreme Gradient Boosting (XGBoost), we introduce a network congestion detection method based on LSH-XGBoost for SSDN. We run testing on open source intrusion detection system: Snort, which was built on SSDN simulator. The promising result proved its capability.

1. Introduction

Railway signal system is one of the most important infrastructures in China, which affects the safety of passengers [1]. Signal Safety Data Network (SSDN) [2] carries the core business of Chinese Train Control System (CTCS). It connects Computer-Based Interlocking (CBI), Train Control Center (TCC), Temporary Speed Restriction Server (TSRS), Radio Block Center (RBC), network management system, etc. Its purpose is to ensure the reliable transmission of safety information between the station and relay station equipment and between the station and the central signal equipment, and its safety will directly affect the traffic safety [3].

Because the SSDN is a double loop network structure, network congestion has a great impact on its availability. In 2009, there was a network storm accident on Shanghai Metro Line 11. The crux of the accident was that the tail fiber of the backbone fiber was wrongly connected due to the illegal construction, resulting in the whole backbone network paralysis [4]. In 2017, Swedish train IT network suffered a wide range of DDoS attacks, resulting in the system collapse and two days of train delay. In 2018, the CTC interface of 6 relay stations in a passenger dedicated line suffered the denial of service, which was caused by the network storm caused by the reversal of the Tick counter of the layer 3 switch [5].
Wu Wencui [6] introduced the concept of hash table into the detection of Goose network storm in intelligent substation. According to the CRC check bit at the end of the good message, this method can judge whether massive repeat Goose message are generated in the current network. Huang Shu et al. [7] designed a double-layer hash table method to detect Goose network storm, which improved the detection of Goose network storm Network storm detection efficiency. But only using hash method can not detect all types of network congestion, because some network storm messages are broadcast storms. In addition, in recent years, for the detection of attacks that can lead to network congestion, researchers focus on the classification of all kinds of attacks by using data mining methods. Keshri A et al. [8] compared with naive Bayes, K-means, support vector machine, cart tree and random forest algorithm, the performance of tree model algorithm is significantly better than other models through KDD99 data set classification problem. Wenli Shang et al. [9] used particle swarm optimization - one class support vector machine (PSO-OCSVM) to detect the intrusion of industrial control system. The model of PSO-OCSVM was simple and accurate, but it could only be used for single classification. Norbert Ádám et al [10] established the neural network intrusion detection system (NNIDS), using DoS attack, Nmap scan, UDP flooding et al. to test NNIDS and achieve good detection results. However, when the data volume is large or the characteristics are complex, the performance of the algorithm is too time-consuming.

To sum up, combined with the possible network congestion in the actual test and the characteristics of the SSDN itself, the existing method has a large calculation cost, and is not enough to detect all possible congestion categories completely. It is not fully applicable to the detection of network congestion in SSDN. Therefore, aiming at the above problems, this paper puts forward a network congestion detection method based on LSH-XGBoost, which can meet the needs of network congestion detection and classification.

2. Overview of network congestion in SSDN

2.1. Basic Structure of SSDN

The basic structure of SSDN is shown in Fig. 1.

SSDN is composed of redundant double ring network through the industrial Ethernet switch equipment of each station and relay station. The double ring network is physically isolated, and the connection between switches adopts special single-mode optical fiber [11]. According to the real-time requirements of services, the transmission delay of each switch's single node shall not exceed 50ms; the self-healing time of single network communication shall not exceed 50ms; the self-healing time of inter network communication shall not exceed 500ms.
2.2. Definition of SSDN Network Congestion

For the SSDN, the concept of network congestion in a broad range is defined: the persistent similar abnormal data in a short period of time in the ring network. In the field and simulation test environment, network congestion will cause communication delay, packet loss, equipment denial of service and other consequences. In this paper, SSDN is based on SICOM3024 layer-2 Switch, and is connected to the looping network through RJ45 interface of 10 / 100Base-T (x) standard.

The possible types of network congestion in SSDN can be divided into three categories: network storm, flooding and scanning (includes port scanning and vulnerability scanning).

2.3. Network Flow Characteristics of SSDN

KDD99 data set [12] is an intrusion detection system (IDS) test data set established by Lincon Laboratory of Massachusetts Institute of technology, which simulates the LAN of the U.S. air force, including four types of attacks: denial of service (DoS), remote to local (R2L), user to root (U2R), and probing. Each flow data is composed of 41 attribute characteristics, and it is the more authoritative data set in IDS field at present. Therefore, this paper divides the flow characteristics into four types: basic data, data content, flow sliding window and time sliding window. The data types include integer (INT), string (STR), boolean (BOOL) and float (FLO), as shown in Table 1.

**Table 1. Flow feature Types of SSDN**

| No. | Lists | Types | Meaning                                      |
|-----|-------|-------|----------------------------------------------|
| C1  | SMAC  | STR   | Source MAC address                           |
| C2  | DMAC  | STR   | Destination MAC                              |
| C3  | SIP   | STR   | Source IP address                            |
| C4  | DIP   | STR   | Destination IP address                       |
| C5  | SPORT | INT   | Source port                                  |
| C6  | DPORT | INT   | Destination port                             |
| C7  | PTYPE | STR   | Protocol type                                |
| C8  | FLAG  | STR   | Connection OK or wrong status                |
| C9  | BYTE  | INT   | Bytes from source                            |
| C10 | BYTE  | INT   | Bytes received from source                   |
| C11 | HOT   | INT   | Access sensitive files                       |
| C12 | NFL   | INT   | Failed logins                                |
| C13 | LOGIN | BOOL  | Login success or not                         |
| C14 | ROOT  | BOOL  | Get root permission or not                   |
| C15 | FC    | INT   | Number of same source and destination IP packets |
| C16 | FCSSIP| INT   | Number of IP packets of the same origin      |
| C17 | FCSDIP| INT   | Number of IP packets of the same destination |
| C18 | FCSDP | INT   | Number of same destination port packets of same origin IP |
| C19 | FCSSP | INT   | Number of same destination IP same origin port packets |
| C20 | FSIZE | INT   | Data size                                    |
| C21 | FTIME | FLO   | Time spent in traffic window                 |
| C22 | TC    | INT   | Number of same source and destination IP packets |
| C23 | TCSSIP| INT   | Number of IP packets of the same origin      |
| C24 | TCSDIP| INT   | Number of IP packets of the same destination |
| C25 | TCSDP | INT   | Number of same destination port packets of same origin IP |
| C26 | TCSSP | INT   | Number of same destination IP same origin port packets |
| C27 | TSIZE | INT   | Data size                                    |
| C28 | TFLOW | FLO   | Time window traffic size                     |
If there are $n$ flow types that may appear in SSDN ($B$ is the type of network congestion flow, with $n-1$ types in total. Normal communication type is defined as $N$). For the $j$-th flow of the $i$-th flow type $FLOW_{i,j}$, we define its flow characteristic value as $C_{i,j}(1), C_{i,j}(2), ... , C_{i,j}(m)$, class FLAG is $1, 2, ... , n-1, n$.

Then the final flow characteristic vector is shown in Table 2.

|       | $C_1$ | $C_2$ | ... | $C_n$ | Flag |
|-------|-------|-------|------|-------|------|
| $B_1$ | $C_{1,j}(1)$ | $C_{1,j}(2)$ | ... | $C_{1,j}(m)$ | 1    |
| $B_2$ | $C_{2,j}(1)$ | $C_{2,j}(2)$ | ... | $C_{2,j}(m)$ | 2    |
| ...   | ...   | ...   | ... | ...   |      |
| $B_{n-1}$ | $C_{n-1,j}(1)$ | $C_{n-1,j}(2)$ | ... | $C_{n-1,j}(m)$ | $n-1$ |
| $N$   | $C_{n,j}(1)$ | $C_{n,j}(2)$ | ... | $C_{n,j}(m)$ | $n$  |

3. Judgment of Network Congestion Based on LSH

3.1. Local Sensitive Hash Algorithm

LSH [13] uses hash conflict to speed up the retrieval process, and drops similar or close data into the same hash bucket through the approximate rate of hash function, which is mainly used for fast approximate search of massive data (such as similar text, image, video, etc.). The LSH algorithm based on Hamming distance is simple in steps and low in algorithm complexity. While the data packets in the network are in binary form, so Hamming distance LSH has good performance in processing massive congestion data, which is suitable for the high real-time detection requirements in the SSDN.

LSH is widely used in $(R, c)$-NN query problems. Its definition is as follows [14]:

For any two points $v, q$ in $S$ space, if the hash function family $H=\{h:S\rightarrow U\}$ satisfies the following two conditions, then $H$ is called $(r_1, r_2, p_1, p_2)$ sensitive.

If $dist(p, q) \leq r_1$, then $P_{r_1}[h(q) = h(v)] \geq p_1$

(1)

If $dist(p, q) \geq r_2$, then $P_{r_2}[h(q) = h(v)] \leq p_2$.

(2)

Where, $D$ is the distance between point $P$ and $Q$, $s$ is the element domain, and satisfies $r_1 < r_2, p_1 > p_2$.

3.2. LSH Function based on Hamming Distance

For binary sequences $a_1$ and $a_2$ whose length is $K$, the Hamming distance between them is defined as follows:

$$\text{Hamming}(a_1, a_2) = \sum_{i=1}^{K} a_1[i] \neq a_2[i]$$

(3)

That is, the number of different bits in the corresponding position. The smaller the Hamming distance between the two binary sequences, the higher the similarity.

Given a family of LSH function $H$, for any $h \in H$, there are:
Where \(a(i,r)\) is the \(r\)-th bit of the \(i\)-th binary sequence. If the Hamming distance of two binary sequences \(p, q\) is \(d\), \(n\) is the length of the sequence, and \(m\) is the number of bits taken, then the probability of entering the same Hash bucket through LSH function is \(\frac{C^m_{n-d}}{C^n_{m}}\) (which \(r\) is a positive integer with uniform distribution), the Hash family \(h\) is \((r_1, r_2, C^m_{n-r_1}, C^m_{n-r_2}, \ldots, C^m_{n-m})\) sensitive.

3.3. Network Congestion Judgment Steps
The steps of network congestion judgment based on LSH are as follows:

1. LSH algorithm initialization. Set reasonable upper limit of hash bucket message \(L\), number of binary bits \(m\) and \(m\) binary bit values \(r_1, r_2, \ldots, r_m\);
2. Flow decoding. The flow data of SSDN within \(t\) seconds before the current time is stored in memory for flow decoding, and the original data packet is decoded into binary sequence \(B_{i,j}\);
3. Network congestion judgment. Parameters \(B_{i,j}, m\) and \(r_1, r_2, \ldots, r_m\) are input into LSH algorithm, and the algorithm outputs message quantity \(H_1, H_2, \ldots, H_{2^m}\) in \(2^m\) hash buckets. If there is an item larger than \(L\) in \(H_1, H_2, \ldots, H_{2m}\), then we determine that there is network congestion in SSDN.

4. Classification of Network Congestion based on XGBoost
The design of network congestion detection system can not affect the availability of SSDN, so we try to improve the TPR of the method and reduce the FPR and FNR. At the same time, due to the real-time requirements of SSDN, the classification algorithm should have good efficiency in mass congestion data. So we use XGBoost [15] to meet the requirements of network congestion classification in SSDN.

4.1. XGboost Algorithm
Build a cart tree integration model for sample set \(\{(x_i, y_i)\}_{i=1}^{n}|D| = n, x_i \in R^m, y_i \in R\), which is expressed as:

\[
\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in F
\]

Where \(F = \{f(x = \omega(q, x)) | q : R^m \rightarrow T, \omega \in R^T \}\) is the set of all CART trees, \(t\) is the number of leaf nodes in CART trees, \(q\) is the leaf node information mapped to each cart tree, the structure \(q\) of each CART tree is independent of the leaf weight \(\omega\), and \(x_i\) is the feature vector. Then the model objective function is as follows:

\[
obj(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)
\]

Where \(l\) is the loss function and \(\Omega\) is the regular term, which is used to measure the complexity of the model. It is defined as:

\[
\Omega(f) = \gamma T + \frac{1}{2} \lambda \|W\|^2
\]
Add training is used to the model to heuristic optimize the objective function. First, optimize \((t-1)-th\) CART tree, then optimize \(t-th\) CART tree, until \(K-th\) CART tree is optimized, as shown in the following formula:

\[
\hat{y}_i^{(0)} = 0 \\
\hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
\vdots \\
\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\]  

(8)

Where \(\hat{y}_i^{(t)}\) is the predicted value after \(t\) times addition training, the objective function of the \(t-th\) CART tree and its second-order Taylor expansion are as follows:

\[
obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)}) + f_t(x_i) + \Omega(f_t) + \text{constant}
\]

\[
\approx \sum_{i=1}^{n} [l(y_i, \hat{y}_i^{(t-1)}) + g_t f_t(x_i) + \frac{1}{2} h_t f_t^2(x_i)]
\]

\[
+ \Omega(f_t) + \text{constant}
\]

(9)

Where \(g_t = \frac{\partial}{\partial \hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})\), \(h_t = \frac{\partial^2}{\partial \hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})\), \(\text{constant}\) is the complexity of the first \(t-1\) CART tree. Remove the constant term from the objective function and expand the regularization term:

\[
obj^{(t)} \approx \sum_{i=1}^{n} [g_{j_i} \omega_{q_i(x_i)} + \frac{1}{2} h_{j_i} \omega_{q_i(x_i)}^2] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} \omega_j^2
\]

\[
= \sum_{i=1}^{T} \left( \sum_{i \in j_i} g_j \omega_j + \frac{1}{2} \left( \sum_{i \in j_i} h_j + \lambda \right) \omega_j^2 \right) + \gamma T
\]

\[
= \sum_{j=1}^{T} \left[ G_j \omega_j + \frac{1}{2} (H_j + \lambda) \omega_j^2 \right] + \gamma T
\]

(10)

Where \(G_j = \sum_{i \in j_i} g_i\), \(H_j = \sum_{i \in j_i} h_i\). For the deterministic structure \(Q(x)\) of the \(t-th\) CART tree, \(G_j\) and \(H_j\) are determined, and the weight \(\omega_j\) of each leaf node is independent of each other. So we can regard \(obj(T)\) as a quadratic function about \(\omega_j\). Then the optimal value \(\omega_j^*\) of the \(j\)-th leaf node and the minimum value \(obj^*\) of the objective function can be expressed as:

\[
\omega_j^* = -\frac{G_j}{H_j + \lambda}
\]

(11)

\[
Obj^* = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T
\]

(12)
4.2. Network Congestion Classification

4.2.1. Model Performance Evaluation Index. We use confusion matrix as the evaluation standard of network congestion classification effect, as shown in Table 3.

Table 3. Confusion Matrix

| Predicted value is normal | True value is normal | True value is network congestion |
|---------------------------|----------------------|----------------------------------|
| Predicted value is network congestion | TN | FN |
| Predicted value is network congestion | FP | TP |

Among them, true negative (TN) is the number of times that normal flow is correctly detected, false negative (FN) is the number of times that congestion data is wrongly detected as normal, false positive (FP) is the number of times that normal data is wrongly detected as congestion, true positive (TP) is the number of times that blocking data is correctly detected. Define true positive rate $TPR = TP / (TF + TP)$, false negative rate $FNR = FN / (TP + TN)$, false positive rate $FPR = FP / (TN + FP)$.

4.2.2. Data Preprocessing. There are four types of features in $C_1 \sim C_{28}$: string, integer, floating and boolean. So we standardize and normalize the features. Because the MAC address and IP address of the equipment in the signal security data network are fixed and continuous, we take the last two bits of MAC address and IP address as the feature $C_1 \sim C_4$. UDP, TCP, ICMP protocols are marked as 0, 1, 2 in turn, and their values are given to characteristic $C_7$. SF means normal, marked as 0, 10 error types are marked as 1 to 10 in turn, and its value are given to characteristic $C_8$. Then the non boolean features in features $C_1 \sim C_{28}$ are normalized:

$$C_{m_{normal}} = \frac{C_m - C_{m_{min}}}{C_{m_{max}} - C_{m_{min}}}$$

(13)

4.2.3. Network Congestion Classification Steps. The construction steps of the network congestion classification model based on XGBoost are as follows:

(1) XGBoost model training. The normal flow and congestion flow of SSDN are preprocessed to form the training set $FLOW = \{B1, B2, \ldots, Bn-1, N\}$. After the maximum depth of CART tree, learning rate and other parameters are set, we standardize and normalize the features. Then we input the feature into XGBoost model to train the model parameters and output the trained model parameters $F = \{f(x) = \omega_{v \cdot \phi}\}$.

(2) Data preprocessing. Extract the flow features $C_1 \sim C_{28}$ of the flow data in SSDN, standardize the string features, and then normalize the non Boolean features.

(3) XGBoost classsification. The XGBoost model trained in step a) is used to predict the current flow and judge the type of network congestion or mark it as normal communication.

5. Simulation Experiment of Network Congestion Detection
Combined with the low time complexity of LSH and the high classification performance of XGBoost, we build a network congestion detection model based on LSH-XGBoost to realize the network congestion detection of SSDN. The model structure is shown in Fig. 2.
First, we decode the flow data of SSDN and input it into LSH algorithm to determine whether there is congestion in the ring network. If LSH determines the flow as congestion, data preprocessing is performed. Then we use XGBoost algorithm to detect the congestion type of flow data.

5.1. Experimental Environment
The theoretical method of this paper is applied to the simulation platform of SSDN. We use four sicom3024 Industrial Ethernet switches to form a ring network to simulate the SSDN structure of the station and central signal equipment site. Two switches are used to simulate the station network, and each switch is connected to two PCs running simulation CBI and TCC software respectively. One switch is connected to PC running simulation TSRS software. One switch is connected to network management system (the software version of network management system is Kyvision Pro F3.2.26-3). The ports of four switches are mirrored to the network congestion simulation verification system, which is based on LAMP and open-source Snort. The architecture of the simulation platform for SSDN and the verification system are shown in Fig. 3.

Snort is an open-source packet sniffer and an extensible lightweight intrusion detection tool [16]. In this paper, we build Snort framework using LAMP component (including Linux operating system, Apache server software, MySQL database and PHP).
5.2. Experimental Result

We use the flow of the SSDN collected on a certain line in China as the normal data, and attack flow launched by the testers in the simulation platform as abnormal data. The data set is shown in Table IV. The training set is used to train XGBoost algorithm, and the test set is used to verify the effectiveness of LSH-XGBoost algorithm.

Based on the five training sets in Table IV, we use xgboost module in Python to train XGBoost algorithm. The parameters of the model are as follows: CART tree maximum depth as 10, learning rate as 0.05, maximum iterations of weak learner as 200, other parameters are default values.

We carry out Hamming distance LSH operation for different kinds of flow data in the first 1s. The value of $m$ is 6, that is, there are $2^6$ hash buckets in total. The sequence length $L$ is the longest message length in the flow data. $r_1, r_2, ..., r_6$ are integers selected from 1 to $L$ according to the characteristics of normal communication message in SSDN. When the length of data message is less than $L$, 0 is filled in the last bits. The simulation results are visualized with hash bucket number as horizontal axis and message number as vertical axis, as shown in Fig. 4.
Figure 4. LSH Hash Bucket Status in 1s under Different Types of Network Congestion

As shown in the Fig. 4, the five flows are represented by different colors. If the threshold value of a single hash bucket is set as \( L = 100 \) (i.e. the black dotted line in Figure 4), it can be seen that the four types of network congestion exceed the set threshold value in a single or multiple hash buckets, while the normal flow of SSDN is always below the threshold value. Therefore, we verify the effectiveness of LSH algorithm in judgment of network congestion.

We verify LSH-XGBoost algorithm for all test sets, and its accuracy, response time and other indicators are shown in Table V.

As shown in Table V, in the verification of four types of test sets, we found that all data sets meet the requirements of real-time communication of SSDN (i.e. response time is far less than 50ms). The accuracy of data set 3 is less than 95%. The reason is that the scanning flow of Nessus is small and the number of packets falling into the same hash bucket is small in unit time, which leads to LSH algorithm misjudging the scanning flow of Nessus as normal traffic. The detection accuracy of test set 1 and 4 is higher than 99.9%, and the average detection accuracy of all test sets is higher than 98%. The results show that the LSH-XGBoost based network congestion detection method is effective.

Table 4. Data Set Composition

| Data Set | Data Set Type | Component | Flow | Duration | Packets Number |
|----------|---------------|-----------|------|----------|----------------|
| Data Set 1 | Training Set 1 | Network Storm | 60Mb/s | 2.47s | 539107 |
| Data Set 1 | Test Set 1 | Network Storm + Normal | 60Mb/s | 2.47s | 538637+470 |
| Data Set 2 | Training Set 2 | Nmap Scan | 513Kb/s | 41.43s | 92069 |
| Data Set 2 | Test Set 2 | Nmap Scan + Normal | 513Kb/s | 41.43s | 84198+7871 |
| Data Set 3 | Training Set 3 | Nessus Scan | 64Kb/s | 310.18s | 75382 |
| Data Set 3 | Test Set 3 | Nessus Scan + Normal | 64Kb/s | 310.18s | 16448+58934 |
| Data Set 4 | Training Set 4 | UDP Flooding | 10Mb/s | 5.28s | 203150 |
| Data Set 4 | Test Set 4 | UDP Flooding + Normal | 10Mb/s | 5.28s | 202147+1003 |
| Data Set 5 | Training Set 5 | Normal | 50Kb/s | 968.13s | 183358 |

\(^a\): The attacker's host is configured as Windows 7 SP1 64 bit operating system with 4GB memory, and the scanning target is the open port of equipment in SSDN.

\(^b\): The host configuration of the attacker is shown in \(^a\), and the Nessus scanning target is the public vulnerability of the equipment in SSDN.
### Table 5. Experimental Results of LSH-xgboost Algorithm

| Data Set | Component          | Judgment based on LSH | Detection based on XGBoost | Accuracy of the LSH-XGBoost (%) |
|----------|--------------------|-----------------------|-----------------------------|-------------------------------|
|          |                    | Accuracy (%) | Detection Time (s) | FNR (%) | FPR (%) | TPR (%) | Detection Time (s) |                         |
| Test Set 1 | Network Storm + Normal | 99.9815     | 0.005849          | 0       | 0       | 100     | 0.005566          | 99.9815                  |
| Test Set 2 | Nmap Scan + Normal   | 98.6741      | 0.006300          | 0.7234  | 0.96    | 99.2766 | 0.007033          | 97.9602                  |
| Test Set 3 | Nessus Scan + Normal | 96.4400      | 0.006852          | 1.6519  | 1.41    | 98.3481 | 0.008020          | 94.8469                  |
| Test Set 4 | UDP Flooding + Normal| 99.9409     | 0.005916          | 0       | 0       | 100     | 0.006873          | 99.9409                  |
| Average  |                    | 98.7591     | 0.006229          | 0.5938  | 0.5925  | 99.4062 | 0.006873          | 98.1824                  |

*: The detection time is calculated by the time module in python2.7.14. The host is configured as Windows 7 SP1 64 bit operating system, i54200m CPU and 4G memory.

### 6. Conclusion

In this paper, we propose a network congestion detection method based on LSH-XGBoost to detect the possible network congestion in SSDN. First of all, we define the generalized form of network congestion in SSDN, and introduce the flow characteristic form and data source. Then, based on the theory of local sensitive hash algorithm, we build a network congestion judgment model based on LSH. Then, based on XGBoost classification algorithm, we establish a network congestion classification model for SSDN, and analyze the performance indicators of the algorithm in the classification of network congestion. In the end, we propose a method of network congestion detection based on LSH-XGBoost in SSDN, and describe the principle and steps of this method. We use the open-source IDS software snort to verify the effectiveness of the proposed method. We use snort, an open source intrusion detection software, to verify the effectiveness of the proposed method. We use snort, an open-source intrusion detection software, to verify the method. The results show that the LSH-XGBoost based network congestion detection method in SSDN is effective.

### Acknowledgment

We thank the Signal & Communication Research Institute, China Academy of Railway Sciences for providing simulation software for signal equipment. The corresponding author of this paper is Wang Jian. And this paper is supported by Ministry of Science and Technology Key R & D Program (2018YFB1201500).

### References

[1] YAN Fei, TANG Tao. Research and Development of Safety Technology in Signalling System of Rail Transit. China Safety Science Journal, 2005(06): 94-99+113.

[2] China Railway Corporation. Technical specification for safety data network of high speed railway signal system V3.0. Beijing: China Railway Press, 2014

[3] LI Saifei, YAN Lianshan, GUO Wei, et al. SD-SSDN: Software-defined Signal Safety Data Network for High-speed Railway Systems. Journal of the China Railway Society, 2018, 40(12): 85-96.

[4] GONG Renshu. Introduction of DCS Communication System Based on CBTC and Causes of Network Storm and Its Treatment. Railway Signalling & Communication Engineering,
2015,12(6):69-74.

[5] Sun Kai, Zhu Ziheng. Analysis of "Network Storm" in Signal Security Data Network of A Passenger Dedicated Line. Railway Signalling & Communication Engineering, 2019,16(01):85-88.

[6] Wu Wencui. Research on the Network Storm Suppression on Process Level in Smart Substations. 2015.

[7] HUANG Shu, MA Wen-shuang, CHEN Jiong-cong, et al. Research on the network storm monitoring and filtering algorithm in smart substation. Power System Protection and Control, 2013(18):68-72.

[8] Keshri A , Singh S , Agarwal M , et al. DoS attacks prevention using IDS and data mining// International Conference on Accessibility to Digital World. IEEE, 2017:87-92.

[9] Shang W , Zeng P , Wan M , et al. Intrusion detection algorithm based on OCSVM in industrial control system. Security and Communication Networks, 2016, 9(10):1040-1049.

[10] Norbert Ádám, Branislav Madoš, Anton Baláž, et al. Artificial neural network based IDS// IEEE 15th International Symposium on Applied Machine Intelligence and Informatics, 2017:159-164.

[11] LIU Dawei. Analysis of Safety Data Network of CTCS-3 Train Control System. Railway Signalling & Communication Engineering, 2010, 07(6):13-15.

[12] Tavallaee M , Bagheri E , Lu W , et al. A detailed analysis of the KDD CUP 99 data set// IEEE International Conference on Computational Intelligence for Security & Defense Applications. IEEE, 2009:53-58.

[13] Piotr Indyk,Rajeev Motwani,Association for Computing Machinery.Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality//Proceedings of the 30th Annual ACM Symposium on Theory of Computing.1998:604-613.

[14] Mayur Datar,Nicole Immorlica,Piotr Indyk, et al.Locality-sensitive hashing scheme based on p-stable distributions//Proceedings of the Twentieth annual symposium on Computational geometry.2004:253-262.

[15] Chen T, Guestrin C. XGBoost: A Scalable Tree Boosting System. ACM SIGKDD International Conference on Knowledge Discovery&Data Mining, 2016:785-794.

[16] ALNABULSI H, ISLAM M R, MAMUN Q. Detecting SQL Injection Attacks Using SNORT IDS// IEEE. The 3rd Asia-Pacific World Congress on Computer Science and Engineering, November 4-5, 2014, Nadi, Fiji. Piscataway: IEEE, 2015:1-7.