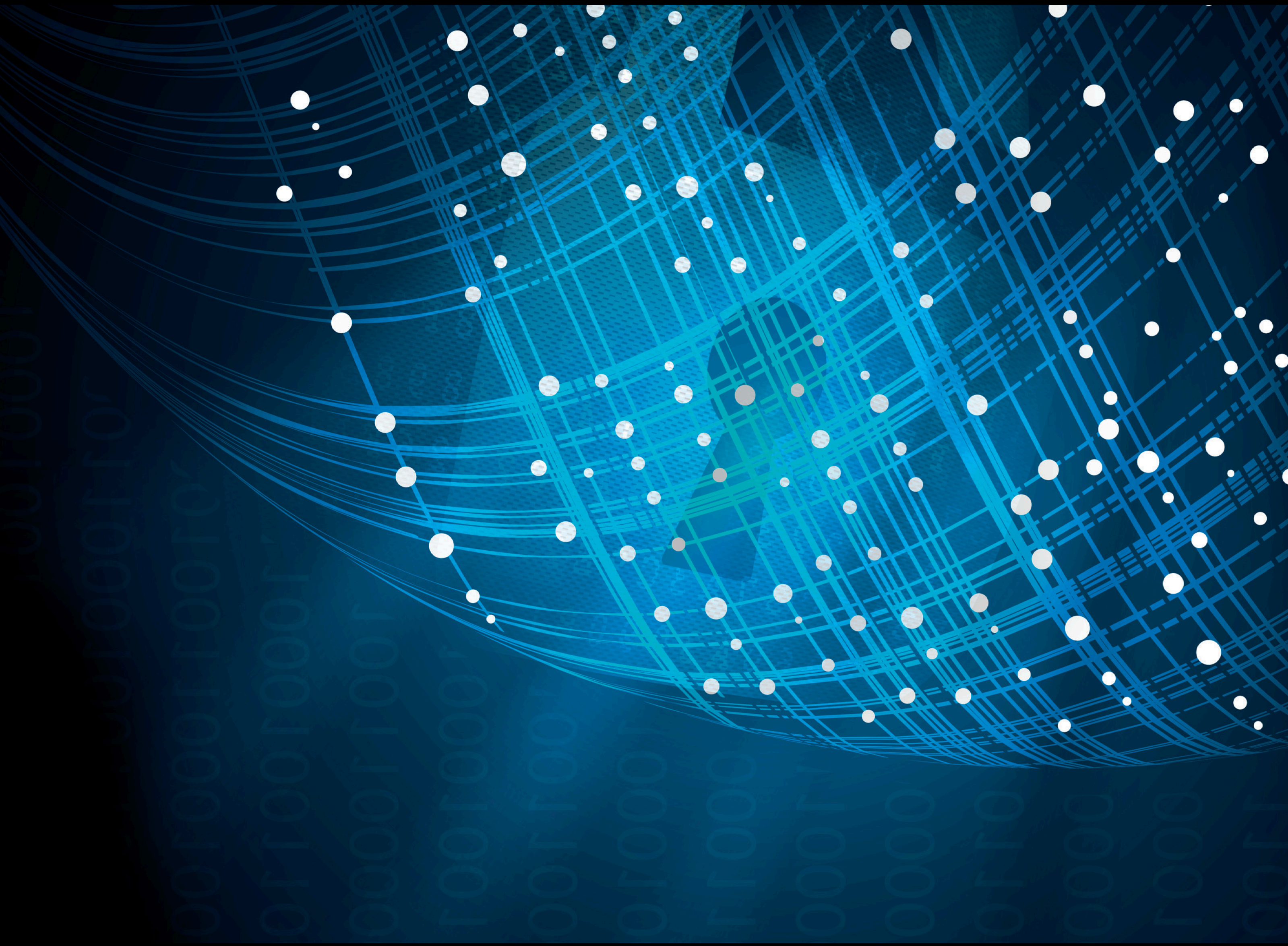
Security and Communication Networks

Machine Learning: the Cybersecurity, Privacy, and Public Safety Opportunities and Challenges for Emerging Applications

Lead Guest Editor: Kehua Guo
Guest Editors: Zhiyuan Tan, Entao Luo, and Xiaokang Zhou



Machine Learning: the Cybersecurity, Privacy, and Public Safety Opportunities and Challenges for Emerging Applications

Security and Communication Networks

Machine Learning: the Cybersecurity, Privacy, and Public Safety Opportunities and Challenges for Emerging Applications

Lead Guest Editor: Kehua Guo
Guest Editors: Zhiyuan Tan, Entao Luo, and Xiaokang Zhou

Chief Editor

Editorial Board

Contents

Hindawi
Security and Communication Networks
Volume 2021, Article ID 9870129, 2 pages
https://doi.org/10.1155/2021/9870129

Editorial

Machine Learning: The Cyber-Security, Privacy, and Public Safety Opportunities and Challenges for Emerging Applications

Kehua Guo,[1] **Zhiyuan Tan**,[2] **Entao Luo**,[3] **and Xiaokang Zhou**[4]

[1]*Central South University, Changsha, China*
[2]*Edinburgh Napier University, Edinburgh, UK*
[3]*Hunan University of Science and Engineering, Yongzhou, China*
[4]*Shiga University, Hikone, Japan*

Correspondence should be addressed to Kehua Guo; guokehua@csu.edu.cn

Received 17 November 2021; Accepted 17 November 2021; Published 3 December 2021

In recent years, as there is continuous advancement of emerging applications such as cyber-physical systems, social networks, e-commerce, and 5G systems, the collection, processing, and analysis of enterprise, government, and personal data have become greatly convenient and widespread, which makes sensitive information more vulnerable to abuses. Therefore, it is urgent to explore secure mechanisms and technologies tailored for emerging applications.

Machine learning (ML) has recently gained a renewed interest as the technology powering it has become more widely available and accessible to organizations of all sizes. Applications using machine learning are being deployed in contexts and for purposes that were not even imaginable a few years ago. From a cybersecurity, privacy, and public safety angle, ML brings about both opportunities and challenges for emerging applications. On the one hand, ML can help interested parties to better protect privacy in challenging situations, improving the state-of-the-art security solutions. On the other hand, ML also presents risks of opaque decision making, biased algorithms, and safety vulnerabilities, challenging traditional notions of privacy protection.

This special issue aims to provide a forum for those from academia and industry to communicate their latest results on theoretical advances and industrial case studies that combine ML techniques, such as reinforcement learning, adversarial machine learning, and deep learning, with significant problems in cybersecurity, privacy, and public safety. Research papers can be focused on offensive and defensive applications of ML to security. Submissions can contemplate original research, serious dataset collection and benchmarking, or critical surveys. Review articles are also welcome.

Potential topics include but are not limited to the following:

(1) Security machine learning modelling and architecture
(2) Secure multiparty computation techniques for machine learning
(3) Attacks against machine learning
(4) Machine learning threat intelligence
(5) Machine learning for cybersecurity
(6) Machine learning for intrusion detection and response
(7) Machine learning for multimedia data security
(8) Machine learning for public safety

After a thorough review process, this special issue has selected a set of eight papers to provide new insights on the abovementioned research areas.

The paper entitled "An Automatic Source Code Vulnerability Detection Approach Based on KELM" by Gaigai Tang, Lin Yang, Shuangyin Ren, Lianxiao Meng, Feng Yang, and Huiqiang Wang proposes to use extreme learning machine (ELM) to effectively improve the iterative training efficiency. In the preprocessing of this framework, they introduce doc2vec for vector representation and multilevel symbolization for program symbolization. Their experimental results show that doc2vec vector representation brings faster training and better generalizing performance than word2vec.

The paper entitled "Malicious URL Detection Based on Improved Multilayer Recurrent Convolutional Neural Network Model" by Zuguo Chen, Yanglong Liu, Chaoyang Chen, Ming Lu, and Xuzhuo Zhang addresses the problem that is hard to fully express the text information of the traditional malicious uniform resource locator (URL) and proposes an improved multilayer recurrent convolutional neural network model based on the YOLO algorithm. Compared with Text-RCNN, BRNN, and other models, their experimental results show that the method detects malicious URLs more quickly and effectively and has high accuracy, high recall rate, and high accuracy.

The paper entitled "Towards Efficient Video Detection Object Super-Resolution with Deep Fusion Network for Public Safety" by Sheng Ren, Jianqi Li, Tianyi Tu, Yibo Peng, and Jian Jiang proposes an efficient video detection object super-resolution with a deep fusion network for public security. By combining the advantages of the pixel-based super-resolution algorithm and the feature space-based super-resolution algorithm, they improve the resolution and the visual perception clarity of the key objects. Extensive experimental evaluations show the efficiency and effectiveness of their method.

The paper entitled "Creating Ensemble Classifiers with Information Entropy Diversity Measure" by Jiangbo Zou, Xiaokang Fu, Lingling Guo, Chunhua Ju, and Jingjing Chen proposes an ensemble classifier generating algorithm to improve the accuracy of an ensemble classification and to maximize the diversity of its component classifiers. Compared with existing classifier methods, it is demonstrated that their method has an obvious lower memory cost with higher classification accuracy.

The paper entitled "Fabric Defect Detection in Textile Manufacturing: A Survey of the State of the Art" by Chao Li, Jun Li, Yafei Li, Lingmin He, Xiaokang Fu, and Jingjing Chen presents a thorough overview of algorithms for fabric defect detection. First, they briefly introduce the importance and inevitability of fabric defect detection towards the era of manufacturing of artificial intelligence. Second, a systematic literature review on defect detection methods is present. Thirdly, the deployments of fabric defect detection algorithms are discussed in their study. They provide a reference for researchers and engineers on fabric defect detection in textile manufacturing.

The paper entitled "An Approach Based on the Improved SVM Algorithm for Identifying Malware in Network Traffic" by Bo Liu, Jinfu Chen, Songling Qin, Zufa Zhang, Yisong Liu, Lingling Zhao, and Jingyi Chen presents an approach for identifying malware in network traffic, called network traffic malware identification (NTMI). Their evaluation results suggest that the NTMI approach can lead to higher accuracy while achieving a lower false positive rate compared with other identification methods. On average, the NTMI approach achieves an accuracy of 92.5% and a false positive rate of 5.527%.

The paper entitled "Representativeness-Based Instance Selection for Intrusion Detection" by Fei Zhao, Yang Xin, Kai Zhang, and Xinxin Niu proposes two instance selection algorithms to handle balanced and imbalanced data problems for intrusion detection. Compared with other algorithms on the benchmark data sets of intrusion detection, their experimental results verify the effectiveness of the proposed instance selection algorithms and demonstrate that the proposed algorithms can achieve a better balance between accuracy and reduction rate or between balanced accuracy and reduction rate.

The paper entitled "Fail-Stop Group Signature Scheme" by Jonathan Jen-Rong Chen, Yi-Yuan Chiang, Wang-Hsin Hsu, and Wen-Yen Lin proposes a fail-stop group signature scheme (FSGSS) that combines the features of group and fail-stop signatures to enhance the security level of the original group signature. Based on the aforementioned objectives, this study proposes three lemmas and proves that they are indeed feasible.

Conflicts of Interest

The editors declare that they have no conflicts of interest regarding the publication of this special issue.

Acknowledgments

In preparing this special issue, we would like to thank all the editors for their support. Furthermore, we would also like to thank Kehua Guo, Entao Luo, Zhiyuan Tan, and Xiaokang Zhou for helping with all cases that needed specific consideration. We thank all reviewers of the manuscripts for this special issue; their efforts have improved the quality of the volume considerably. We also thank the authors for participating in and contributing to this special issue.

Kehua Guo
Zhiyuan Tan
Entao Luo
Xiaokang Zhou

Hindawi
Security and Communication Networks
Volume 2021, Article ID 5566423, 12 pages
https://doi.org/10.1155/2021/5566423

Research Article

An Automatic Source Code Vulnerability Detection Approach Based on KELM

Gaigai Tang,[1,2] Lin Yang,[2] Shuangyin Ren,[2] Lianxiao Meng,[1,2] Feng Yang,[2] and Huiqiang Wang[1]

[1]*School of Computer Science and Technology, Harbin Engineering University, Harbin, China*
[2]*National Key Laboratory of Science and Technology on Information System Security, Institute of System Engineering, Chinese Academy of Military Science, Beijing, China*

Correspondence should be addressed to Huiqiang Wang; wanghuiqiang@hrbeu.edu.cn

Received 12 January 2021; Revised 16 March 2021; Accepted 19 May 2021; Published 16 June 2021

Academic Editor: Xiaokang Zhou

Traditional vulnerability detection mostly ran on rules or source code similarity with manually defined vulnerability features. In fact, these vulnerability rules or features are difficult to be defined accurately, which usually cost much expert labor and perform weakly in practical applications. To mitigate this issue, researchers introduced neural networks to automatically extract features to improve the intelligence of vulnerability detection. Bidirectional Long Short-term Memory (Bi-LSTM) network has proved a success for software vulnerability detection. However, due to complex context information processing and iterative training mechanism, training cost is heavy for Bi-LSTM. To effectively improve the training efficiency, we proposed to use Extreme Learning Machine (ELM). The training process of ELM is noniterative, so the network training can converge quickly. As ELM usually shows weak precision performance because of its simple network structure, we introduce the kernel method. In the preprocessing of this framework, we introduce doc2vec for vector representation and multilevel symbolization for program symbolization. Experimental results show that doc2vec vector representation brings faster training and better generalizing performance than word2vec. ELM converges much quickly than Bi-LSTM, and the kernel method can effectively improve the precision of ELM while ensuring training efficiency.

1. Introduction

As software becomes more and more complicated, software vulnerabilities caused by design flaws and implementation errors become an inevitable problem in engineering [1]. According to statistics released by the Common Vulnerabilities and Exposures (CVE) [2] and National Vulnerability Database (NVD) [3], the number of software vulnerabilities has increased from 1600 to nearly 100000 since 1999 [4]. Software systems containing these vulnerabilities will face serious security risks.

On the one hand, existing vulnerability detection techniques are mostly driven by rules [5–10] and code similarity metrics [11, 12]. Vulnerability detection rules are usually defined by experienced experts. The performance of these methods is limited by the experience of experts. Generally, the features of software vulnerabilities are very difficult to be described accurately, which leads to the corresponding detection rules which are also difficult to be defined accurately and completely.

These problems inspired researchers to propose automatic vulnerability detection (source code level). Neural networks show great potential [13–17]. Neural networks can automatically extract complex features from input data, avoiding the problems of high cost, instability, and incompleteness of manually constructing features and empirically defining rules. VulDeePecker [16] utilized Bi-LSTM [18] for software vulnerability detection. Zhen Li et al. [17] discussed the performance of different neural networks on vulnerability detection separately, namely, MLP, CNN, LSTM, and Bi-LSTM. All of the above neural networks train the detection model with an iterative training mechanism, which usually costs a lot of time. To solve this problem, we introduce ELM [19], which trains the detection model with a noniterative training mechanism. In order to improve precision performance, we then introduce the kernel method.

On the other hand, there are two most classical data preprocessing methods in neural network-based automatic vulnerability detection, namely, vector representation and program symbolization. The most common vector representation method is word2vec [20], which can vectorize the software codes into form of vector (variable length) as the input of neural network. However, word2vec usually requires additional work to further preprocess the output vector (e.g., padding zeros). The final vectors usually with large dimension can heavily affect the training efficiency of detection model. Moreover, word2vec may also lose important semantic information of the source codes, which can affect the precision of detection model. As for program symbolization, the normal way is to symbolize the variables and user-defined functions in the source code at the same time [16,17], which can be seen as a single symbolization level of 2. This idea ignores to consider the influence of multiple symbolization levels on performance of vulnerability detection model.

To alleviate the above problems, we propose a multilevel symbolization method for symbolic representation and introduce doc2vec [21] for vector representation. In detail, we first obtain symbolic representations of the source codes related to vulnerabilities through three symbolizations. Using three levels of symbolization can significantly reduce the noise introduced by irrelevant information of vulnerable codes. Then, we use doc2vec to automatically transform symbolic representation of source codes to corresponding vector representation. Compared to word2vec used in [16], we found that doc2vec is more suitable for modeling vector representation because it can not only transform source codes with arbitrary length into a fixed-length feature representation but also grasp the semantic information of source codes better. These advantages are helpful to improve the precision and training efficiency of vulnerability detection model.

The rest of this paper is organized as follows. Section 2 discusses the work related to automatic detection of software vulnerability. Section 3 describes the details of the proposed automatic software vulnerability detection method. Section 4 gives the details of experimental environment and parameter configuration, experimental results, and corresponding analysis. The conclusions and future works are presented in Section 5.

2. Related Work

2.1. Vulnerability Detection Techniques. Existing classical vulnerability detection techniques range from making use of manually defined features [5–10] to code similarity metrics [11, 12]. However, there are several primary flaws among them. First, the effort for defining vulnerability features is error-prone and manual labor consuming. Second, the features can hardly be integral and usually contain only partial information about the vulnerabilities, which may lead to high false-positive and false-negative rates [16]. Moreover, the application of the code similarity method is limited to vulnerabilities caused by code clones.

Vulnerability detection with traditional machine learning techniques such as Decision Tree [22] and Support Vector Machine (SVM) [23] mainly extracts vulnerability features from preclassified vulnerabilities. However, vulnerability detection patterns based on this type of feature are usually available for specific vulnerabilities. In the paper by Boris Chernis [24], both simple text features (e.g., character count, character diversity, and maximum nesting depth) and complex text features (e.g., character n-grams, word n-grams, and suffix trees) are extracted from the source codes and analyzed by using the naive Bayes classifier. Experimental results show that simple features performed unexpectedly better by comparing with the complex features.

Neural networks can learn complex vulnerability features automatically. Zhen Li [16] presented a vulnerability detection system VulDeePecker based on deep learning, which initiates the study of using deep learning for vulnerability detection. VulDeePecker collects the samples by first extracting code gadgets from the buggy programs and then transforming them into the vector representations using word2vec. The detection model is designed based on Bi-LSTM. Siqi Ma [13] proposed a tool called VuRLE for automatic detection and repair of vulnerabilities. VuRLE uses the context patterns to detect vulnerabilities and customizes the corresponding edit patterns to repair them. Jacob A. Harer [14] implemented various machine learning models for detecting bugs that can lead to security vulnerabilities in C/C++ code. Specifically, they used features derived from the build process and the source code. Rebecca L. Russell [15] developed a vulnerability detection tool based on deep feature representation learning that can directly interpret the parsed source codes. The source codes are firstly transformed into tokens and then embedded as vectors for both CNNs and Recurrent Neural Networks (RNNs). Zhen Li [25] proposed a systematic framework by using deep learning to detect vulnerabilities that combined syntax-based, semantics-based, and vector representations (SySeVR). SySeVR can accommodate syntax and semantic information pertinent to vulnerabilities. The source codes are successively represented by syntax-based, semantics-based, and vector representations. Zhen Li [17] performed a quantitative evaluation of the impacts of different factors (e.g., data dependency and control dependency) on the effectiveness of neural network-based vulnerability detection techniques. Zhen Li [26] presented VulDeeLocator, a deep learning-based fine-grained vulnerability detector. It leverages intermediate code to capture semantic information that cannot be conveyed by source code-based representations and presents a new idea of granularity refinement. Xin Li [27] proposed an automated and intelligent vulnerability detection method in source code based on the minimum intermediate representation learning. The sample in the form of source code is first transformed into a minimum intermediate representation; then, it is transformed into a real value vector through pretraining on an extended corpus. The vector is fed to three concatenated convolutional neural networks to obtain high-level features of vulnerability.

2.2. Preprocessing Method. The commonly used preprocessing methods for automatic source code vulnerability detection are program symbolization and vector representation. Zhen Li [16, 25] first maps variable names to symbolic names (e.g., "$V1$" and "$V2$") in a one-to-one fashion, then maps function names to symbolic names (e.g., "$F1$" and "$F2$") in a one-to-one fashion, and finally uses word2vec to perform vector representation. Gustavo Grieco [28] uses word2vec to preprocess the dynamic features of source codes since it was successfully used in a variety of text mining applications. Savchenko [29] proposed a system for vulnerability detection based on deep learning approach, which performs the following steps: source code preprocessing, AST creation, code gadget extraction, and code gadget vectorization using word2vec.

2.3. Kernel Method. The kernel method is often used to solve the linear indivisibility problems. Qin-Qin Tao [30] proposed a locality-sensitive support vector machine using kernel combination (LS-KC-SVM) algorithm, which solved the large appearance variations due to some real-world factors on face detection. Liang [31] proposed an SVM-based method combining with the deep quasilinear kernel (DQLK) learning for large-scale image classification. It could train SVM on a large-scale dataset with less memory space and less training time. Zhang [32] developed a least-squares (LS) SVM-based identification scheme, where the system parameters were estimated in a reproducing kernel Hilbert space. It can effectively solve the issue that LS results in low accuracy in ill-conditioned scenarios. Lu Li [33] proposed the AdaBoost-WCKELM made of ELM, AdaBoost, and composite kernel method, which derived a good improvement in HSI classification accuracy.

3. The Methodology

Figure 1 is an overview of the proposed automatic source code vulnerability detection system using enhanced ELM on the source code level. Starting with the dataset in form of code gadget, it then obtains symbolic representation of each code gadget using multilevel symbolization. Next, it transforms the symbolic representations into vector representations with a low-dimension using doc2vec. Finally, it applies enhanced ELM neural networks to train the detection model. As for testing, code gadget is firstly preprocessed successively through multilevel symbolization and doc2vec, and then the vector representations of them are input to detection model to get the detection results. In the subsequent sections, we give the details of the main components of this system.

3.1. Symbolic Representation. A code gadget is composed of several program statements (i.e., lines of code), which are semantically related to each other in terms of data dependency or control dependency [16]. It can be further transformed into a form of symbolic representation using symbolization. The symbolic representation is then collected as a corpus for training the vector representation tool, such as doc2vec.

The benefit of symbolic representation is that it can result in higher training effectiveness by further reducing the length of code gadget. In symbolization, vulnerability features of each code gadget such as local variables, user-defined functions, and data types are transformed into short and fixed-length symbolic presentations, where the same features are mapped to the same symbolic presentation. In this work, we deploy three symbolization types that are shown as follows:

(i) Function calls symbolization (F): User-defined function names are symbolically represented as FN. This symbolization type is assigned the priority because vulnerability is mostly caused by improper utilization of library/API function calls. Symbolization on user-defined functions can improve the Signal-Noise Ratio (SNR) of library/API function in vulnerability information.

(ii) Variable symbolization (V): Variable names including parameters and local variables are symbolically represented as VN. In practice, the variables account for a large proportion of the codes.

(iii) Data type symbolization (T): Data types of variable and user-defined function are symbolically represented as TN. It has the least priority since many data types are not related to vulnerability information.

The symbol N mentioned above in symbolization is a number which represents the index of the first occurrence of the feature while noting that multiple functions may be mapped to the same symbolic name when they appear in different code gadgets. Moreover, all the symbolization types will reserve keywords of C/C++ language.

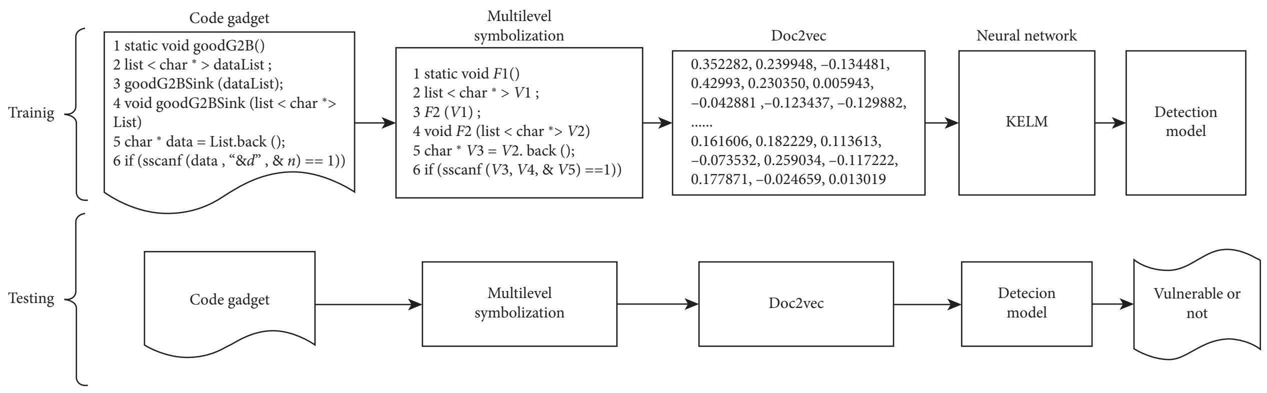
We build a multilevel symbolization mechanism according to the priority of symbolization shown in Table 1. Level 2 includes two symbolization groups, namely, $F+V$ and $F+T$. This is because symbolizations V and T may have different effects on SNR of vulnerability information in different datasets.

We take Sample 0 as an example to show how the symbolization works, where the symbolization group $F+V$ is chosen from level 2. From Figure 2, we can observe that there are 2 user-defined functions, 5 variables, and 2 data types in Sample 0.

(i) In level 1, the two user-defined functions are symbolically represented as $F1$ and $F2$.

(ii) In level 2, the five variable names are symbolically represented as $Vi, i \in [1,5]$.

(iii) In level 3, the two data types are symbolically represented as $T1$ and $T2$.



```
static void goodG2B()
list < char * > dataList ;
goodG2BSink (dataList);
void goodG2BSink (list < char *> List)
char * data = List.back ();
if (sscanf (data , “&d” , & n) == 1))
```

```
static void F1()
list < char * > V1 ;
F2 (V1) ;
void F2 (list < char *> V2)
char * V3 = V2. back ();
if (sscanf (V3, V4, & V5) ==1))
```

```
0.352282, 0.239948, −0.134481,
0.42993, 0.230350, 0.005943,
−0.042881 ,−0.123437, −0.129882,
......
0.161606, 0.182229, 0.113613,
−0.073532, 0.259034, −0.117222,
0.177871, −0.024659, 0.013019
```

Figure 1: Overview of the proposed automatic source code vulnerability detection system using KELM.

Table 1: Multilevel symbolization.

Symbolization level	Symbolization group
Level 1	F
Level 2	$F+V$; $F+T$
Level 3	$F+V+T$

```
static void goodG2B()
list < char* > dataList;
goodG2BSink(dataList);
void goodG2BSink(list < char* > List)
char * data = List.back();
if(sscanf(data,“&d”, &n) == 1))
```

(a)

```
static void F1()
list < char* > dataList;
F2(dataList);
void F2(list < char* > List)
char * data = List.back();
if(sscanf(data,“&d”, &n) == 1))
```

(b)

```
static void F1()
list < char* > V1;
F2(V1);
void F2(list < char* > V2)
char * V3 = V2.back();
if(sscanf(V3,V4, &V5) == 1))
```

(c)

```
T1 F1()
list < T2* > V1;
F2(V1);
void F2(list < T2* > V2)
T2 * V3 = V2.back();
if(sscanf(V3,V4, &V5) == 1))
```

(d)

Figure 2: An example of multilevel symbolization of source code. (a) Sample 0. (b) Level 1 F. (c) Level 2 $F+V$. (d) Level 3 $F+V+T$.

As a result, through three levels of symbolization, Sample 0 is gradually simplified to a generalized symbolic representation, which can effectively characterize different manifestations of the same vulnerability.

3.2. Vector Representation. Since the neural network can only accept vector as input, the symbolic representation of source code needs to be further converted to the vector representation. Currently, the most popular vectorization methods are word2vec [34] and doc2vec [35].

Compared with the one-hot representation, a high-dimensional and sparse representation method, word2vec, outputs a low-dimensional and dense vector representation, which is conducive to improving training efficiency and precision of the model, making it widely used for vulnerability detection recently [14, 16, 17]. However, there is a drawback of word2vec; that is, it ignores the influence of word order that relates to information of a sentence or a document.

doc2vec was proposed in [35], and the authors proposed the unsupervised algorithm called Paragraph Vector that can learn fixed-length feature representation from texts with arbitrary length, ranging from a sentence to a document. Moreover, the Paragraph Vector can memorize the topic of the paragraph, which makes it be able to better extract global features than word2vec.

Given the fact that word2vec converts word to vector representation in a one-to-one fashion, thus, the length of the converted vector varies with the length of the input text. To satisfy the neural network requirement of input with a fixed length, the vector generated by word2vec needs to be further processed to obtain the corresponding fixed-length form. Different from word2vec, doc2vec can directly output fixed-length vectors from input texts with arbitrary length. Furthermore, doc2vec can also grasp more semantic information from the context of input text than word2vec. In summary, doc2vec shows great potential in source code vector representation.

3.3. Neural Network Model. ELM is a special type of feedforward neural network with the noniterative training mechanism, which was proposed by Huang et al. in the 1990s [19]. Unlike traditional neural networks, which use gradient descent techniques to iteratively fine-tune all the parameters of the model, ELM randomly assigns values to some parameters according to certain rules and keeps these parameters frozen throughout the training process, while other parameters are calculated by the least square method. In other words, the training mechanism of ELM is noniterative, which can bring it much faster training speed than conventional neural networks on some tasks with relatively large data scale. Here, we take ELM with a single hidden layer network

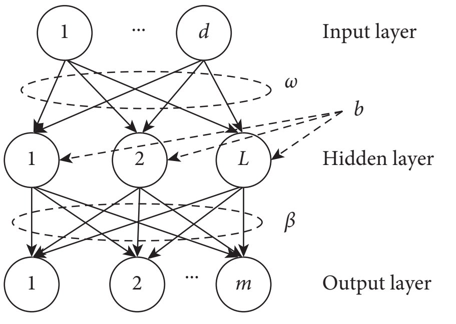


Figure 3: A typical ELM with a single hidden layer network structure.

structure as an example to introduce its training mechanism. The network structure of ELM is shown in Figure 3.

3.3.1. ELM. In Figure 3, d, L, and m refer to the number of the input layer neurons, the hidden layer neurons, and the output layer neurons, respectively. ω is the input weights connecting the input layer to the hidden layer, $\mathbf{b}$ is the thresholds of the hidden layer neurons, and β is the output weights connecting the hidden layer to the output layer. ω and $\mathbf{b}$ are generated randomly from the range (−1, 1) and (0, 1) under a uniform distribution. They are kept frozen throughout the training process of the model.

Given a training data set $D = \{(\mathbf{x}_i, \mathbf{t}_i)|\mathbf{x}_i \in R^d, \mathbf{t}_i \in R^m\}, i = 1, 2, \ldots, N$, the ELM model can be represented as

$$\begin{aligned}
&\mathbf{H\beta} = \mathbf{T},\\
&\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} h(\mathbf{x}_1) & \ldots & h_L(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ h_1(\mathbf{x}_N) & \ldots & h_L(\mathbf{x}_N) \end{bmatrix},\\
&\mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix} = \begin{bmatrix} t_{11} & \ldots & t_{1m} \\ \vdots & \vdots & \vdots \\ t_{N1} & \ldots & t_{Nm} \end{bmatrix},
\end{aligned} \tag{1}$$

where $\mathbf{T}$ is the expected output matrix and $\mathbf{H}$ is the hidden layer output matrix. $\mathbf{h}(\mathbf{x}_i) = \sum_{j=1}^{L} g(\omega_j \cdot \mathbf{x}_i + b_j), i = 1, 2, \ldots, N$, which is the output vector of the hidden layer with respect to the input $\mathbf{x}_i$. $g(\cdot)$ is the activation function of the ELM. And $\omega_j \cdot \mathbf{x}_i$ denotes the inner product of the input weights and the features of the ith training sample. The output weights β can be obtained by

$$\boldsymbol{\beta} = \mathbf{H}^{+}\mathbf{T} = \begin{cases} \mathbf{H}^T\left(\lambda\mathbf{I} + \mathbf{HH}^T\right)^{-1}\mathbf{T}, & \text{when } N \leq L, \\ \left(\lambda\mathbf{I} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T}, & \text{when } N \geq L, \end{cases} \tag{2}$$

where $\mathbf{H}^{+}$ refers to the Moore – Penrose generalized inverse of $\mathbf{H}$, L refers to neuron number of hidden layers, $\mathbf{I}$ refers to an N identity matrix, and λ refers to a regularization factor with a value between [0,1].

The ELM output function is

$$f(\mathbf{x}) = \begin{cases} \mathbf{h}(\mathbf{x})\mathbf{H}^T\left(\lambda\mathbf{I} + \mathbf{HH}^T\right)^{-1}\mathbf{T}, & \text{when } N \leq L, \\ \left(\lambda\mathbf{I} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{h}(\mathbf{x})\mathbf{T}, & \text{when } N \geq L. \end{cases} \tag{3}$$

The optimization objective of the ELM model can be expressed as

$$\min\left(\sum_{i=1}^{N}\left\|f(\mathbf{x}_i) - \mathbf{t}_i\right\|^2\right), \tag{4}$$

where $f(\mathbf{x}_i)$ and $\mathbf{t}_i$ refer to the predictive label and the real label of the ith sample, respectively.

3.3.2. KELM. Kernel method is an effective way to solve the nonlinear problems by mapping the data to high-dimensional space so that the nonlinear problem can be transformed into a linear problem. With the combination of kernel method, there are two benefits compared with conventional ELM. For one thing, it solves the problem that the number of hidden layer nodes in conventional ELM depends on manual setting, which shows better stability [36]. For another thing, the kernel function maps the data to the high-dimensional space, and the distribution of the data in the transformed space is very smooth. In fact, the smooth new data make the classification problem easier, so the model can show better effectiveness. Radial Basis Function (RBF) is the preferred kernel function in our experiments because it has only one hyperparameter which simplifies the model configuration and training cost. RBF kernel function can be expressed as

$$K(\mathbf{x}, \mathbf{y}) = e^{-\gamma\|\mathbf{x}-\mathbf{y}\|^2}, \tag{5}$$

where $\mathbf{x}$ and $\mathbf{y}$ represent the samples, γ represents the unique hyperparameter of Gaussian kernel function, and $\|\mathbf{x} - \mathbf{y}\|$ denotes the norm of vectors.

The kernel matrix for ELM can be defined as [37]

$$\begin{aligned}
\mathbf{\Omega}_{\text{ELM}} &= \mathbf{H}^T\mathbf{H},\\
\mathbf{\Omega}_{\text{ELM}_{ij}} &= \mathbf{h}(\mathbf{x}_i) \cdot \mathbf{h}(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j).
\end{aligned} \tag{6}$$

And we can revise equation (2) when $N \geq L$ as

$$\boldsymbol{\beta} = (\lambda\mathbf{I} + \mathbf{\Omega})^{-1}\mathbf{H}^T\mathbf{T}, \tag{7}$$

and then, the ELM output function (3) can be as follows:

$$\begin{aligned}
f(\mathbf{x}) &= \left(\lambda\mathbf{I} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{h}(\mathbf{x})\mathbf{T}\\
&= (\lambda\mathbf{I} + \mathbf{\Omega})^{-1}\begin{bmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^T \mathbf{T}.
\end{aligned} \tag{8}$$

From equation (8), we can find that ELM combined with the kernel method can avoid the problem that the number of hidden layer nodes in conventional ELM depends on manual setting.

4. Experiment and Evaluation

The goal of our work is to construct an automatic software vulnerability detection model with both superior precision and efficiency. To be specific, we investigate the following questions in experiments:

(i) Question1 ($Q1$): How differently do neural network models perform on vulnerability detection?

(ii) Question2 ($Q2$): How differently do vector representation methods affect the performances of neural networks? Specifically, does doc2vec outperform word2vec on vulnerability detection?

(iii) Question3 ($Q3$): What are the effects of different symbolization on the performances of neural networks?

4.1. Experiment Setting and Implementation

4.1.1. Dataset. In our experiments, we include the following three datasets from [16]. Each sample is a piece of source code with known vulnerabilities. Table 2 shows the number of samples (i.e., source code files) in each dataset.

(i) BE-ALL includes samples with buffer error vulnerabilities (CWE-119) and ALL library/API function calls.

(ii) RM-ALL includes samples with resource management error vulnerabilities (CWE-399) and ALL library/API function calls.

(iii) HY-ALL includes samples with hybrid buffer error vulnerabilities (CWE-119), resource management error vulnerabilities (CWE-399), and ALL library/API function calls.

Each dataset is partitioned into two parts with a proportion of 80% and 20%, where the larger part is for training and the other part is for testing. Each sample in the dataset is in the form of code gadget with a ground truth label.

4.1.2. Evaluation Metrics. In our experiment, we used the indexes mentioned in [38] to evaluate the effectiveness of vulnerability detection model, that is, False Positive Rate (FPR), True Positive Rate (TPR), Precision (P), and F1-measure (F1). The value range of these four indicators is [0, 1]. For FPR, the closer their values are to 0, the better the performance of the model is; for other indicators, the closer their values are to 1, the better the performance of the model is.

The quality of vector representation can be evaluated by Cosine Similarity (cosine) between vectors in the vector space, which can be calculated by the following formula. The range of cosine value is [−1, 1]. The closer the value is to 1 or −1, the more similar the two vectors are.

$$\text{cosine}(A, B) = \frac{A \cdot B}{\|A\|_2 \|B\|_2}, \quad (9)$$

Table 2: Number of samples in each dataset.

Dataset	Code gadgets	Vulnerable code gadgets
BE-ALL	39753	10440
RM-ALL	21885	7285
HY-ALL	61638	17725

where A and B refer to vectors. Given the fact that Cosine Similarity only considers the angle between vectors, so that it can avoid too large output deviation due to different dimension of input vectors. This is the main reason why we choose Cosine Similarity as the evaluation metric of vector representation.

4.1.3. Parameters Setting for Neural Networks. In our experiments, we used two types of neural networks for the vulnerability detection model, namely, Bi-LSTM and ELM. For both, there is only one hidden layer in the network structure. We build the following five configurations. We do not list the configuration of AdaBoost KELM because it is predictable that the calculation of KELM with weight and iteration mechanism is very complex and the efficiency will be greatly reduced.

(i) word2vec with Bi-LSTM ($w + B$), which was used by VulDeePecker

(ii) doc2vec with Bi-LSTM ($d + B$)

(iii) doc2vec with ELM ($d + E$)

(iv) doc2vec with AdaBoost ELM ($d +$ Ada-E)

(v) doc2vec with KELM ($d +$ KE)

We have implemented the CPU versions of Bi-LSTM and ELM, and all the models were trained in the PC environment with CPU. For Bi-LSTM, the batch size, the dropout rate, the number of epochs, and the number of the hidden layer neurons were set to 64, 0.5, 2, and 60, respectively, and the optimizer chosen was Root Mean Square Prop (RMSProp). For ELM, the number of the hidden layer neurons was set to 5000 and the activation function used sigmoidal function. The input weights and the hidden biases of ELM were generated randomly from (−1, 1) and (0, 1), respectively, under a uniform distribution. The details of the parameters' configuration of ELM are given as follows.

To determine which activation function is the best choice for the ELM-based detection model, we implement an experiment to discuss the effectiveness of ELM with five activation functions, respectively. The number of neurons is set to 250 and the dataset is HY-ALL. From the results in Figure 4, we can find that ELM with sigmoidal function outperforms the other activation functions on precision and F1.

In terms of neuron configuration of ELM, we have done several experiments to analyze the effect of a different number of neurons on the precision of ELM as shown in Table 3. Generally speaking, when the number of neurons ranges from 250 to 12000, the precision of ELM gradually increases with the number of neurons increasing, but when the number of neurons is more than or equal to 15000, the

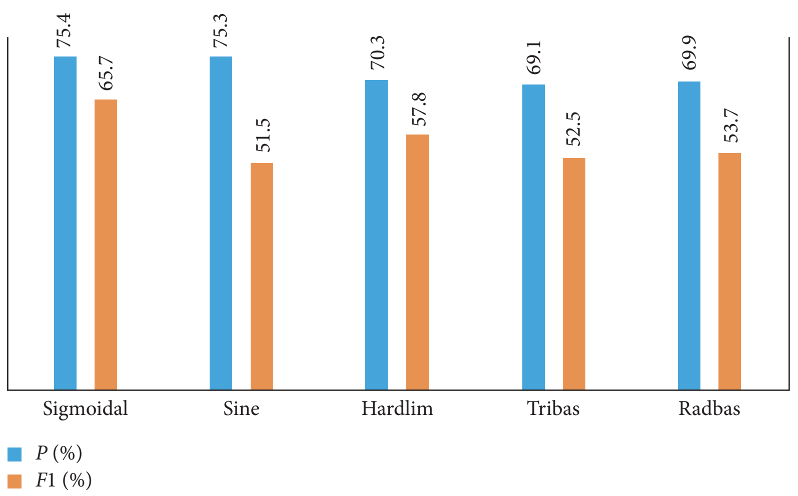


Figure 4: Effect of different activation function on the precision of ELM.

Table 3: Effect of different number of neurons on the precision of ELM.

Neuron number	FPR (%)	TPR (%)	*P* (%)	*F*1 (%)	Training time (s)
250	7.5	58.2	75.4	65.7	0.92
500	6.6	66.1	80.0	72.4	2.52
1000	6.4	72.8	81.9	77.1	7.75
3000	5.4	80.5	85.5	82.9	49.11
5000	4.4	83.6	**88.3**	**85.9**	**128.72**
10000	4.4	85.9	88.6	87.2	496.02
12000	4.3	85.9	88.9	87.4	640.68
15000	4.4	86.6	88.6	87.6	1143.70
20000	4.6	86.7	88.2	87.4	2201.60

precision of ELM begins to decline slowly. In particular, when the number is between 250 and 5000, the precision improvement is more obvious, while the number increases from 5000 to 12000, the precision improvement is slight, nearly 0.3%, and the training time increased by 4 times. Considering the cost-effectiveness of precision improvement and time consumption, we set the number of neurons as 5000.

Kernel function plays a very important role in KELM, which largely determines its precision performance. We collect three commonly used kernel functions to make a comparison experiment. The comparison of the results after fine-tuning is shown in Figure 5. It is clear from the result that RBF shows the best overall performance than the other two kernel functions. Thus, the subsequent KELM-related experiments set the RBF as the kernel function.

4.2. Results and Evaluation

4.2.1. Results for Q1. Regarding the impacts of different neural network models on the performances of vulnerability detection, we evaluate the precision and efficiency of the above five configurations on all datasets. Table 4 shows the effect of different neural network models on vulnerability detection precision, while Table 5 gives the efficiency of different neural network models on vulnerability detection. In the experiments, all three datasets are preprocessed with the symbolization group $F + \mathrm{V}$.

According to the results in Table 4, we analyze them from two aspects: precision comparison of conventional Bi-LSTM and ELM and enhanced effect of conventional ELM using kernel function and AdaBoost method.

Compared with ELM, Bi-LSTM is slightly inferior in RM-ALL, a small-scale dataset, but superior in BE-ALL and HY-ALL, the large-scale datasets. This may be due to the fact that the deep learning model is more suitable for large dataset scenarios. Besides, Bi-LSTM shows lower FPR than ELM on all three datasets, which can be explained by the fact that Bi-LSTM can express the long-term dependency information in the input, while ELM is based on forwarding neural network; it is slightly inferior to Bi-LSTM in the context processing.

For Ada-E, it outperforms conventional ELM on RM-ALL and HY-ALL, which shows the advantage of the ensemble learning, for example, combination enhancement. However, it shows similar P and lower TPR than ELM on RM-ALL, which may be due to the overfitting effect of ensemble learning for high-precision base classifiers. It can be seen that if the base classifier is with very high precision, the final classifier generated by AdaBoost does not always show the higher precision but may be worse if the basic classifier shows high enough precision. For KE, it shows the lowest FPR and the highest P on the three datasets compared with the other five configurations, which benefits from its effective way to solve nonlinear problems through high-dimensional mapping. Besides, it also results in the lowest TPR, but this is acceptable; it is due to the fact that the high false-positive rate is the primary problem of vulnerability detection tools in practical application.

From Table 5, we can find that the configuration $w + B$ performs the longest time for training and detection on HY-ALL, while configuration $d + B$ costs less than 1/30 of configuration $w + B$. It is because the configuration $w + B$ in [16] outputs vectors with a longer dimension of 2500, which

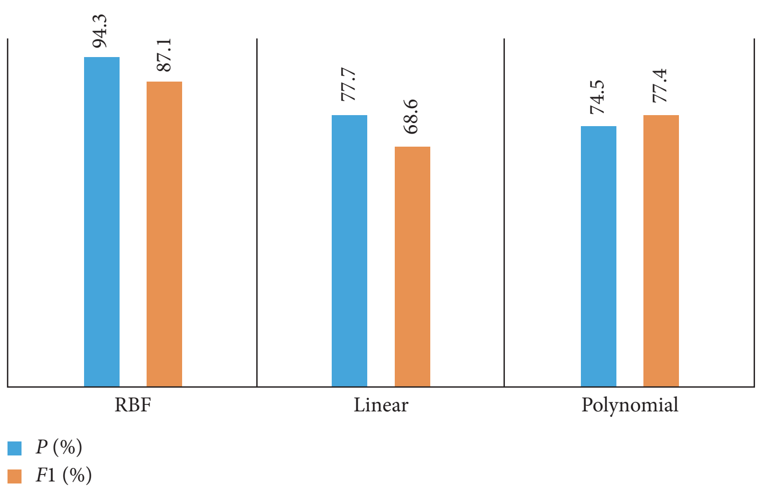


Figure 5: Effect of different kernel function on the precision of KELM.

Table 4: Effect of different neural network models on vulnerability detection precision.

Dataset	Pattern	FPR (%)	TPR (%)	P (%)	$F1$ (%)
BE-ALL	$w+B$	2.9	82.0	91.7	**86.6**
	$d+B$	3.9	83.1	88.1	85.5
	$d+E$	4.4	81.9	86.8	84.3
	$d+$ Ada-E	3.9	82.7	88.1	85.3
	$d+$ KE	1.8	78.7	**93.8**	85.6
RM-ALL	$w+B$	2.8	95.3	94.6	**95.0**
	$d+B$	3.8	90.3	91.9	91.1
	$d+E$	3.9	92.4	94.9	92.1
	$d+$ Ada-E	2.8	83.8	93.4	88.3
	$d+$ KE	1.1	82.7	**97.4**	89.5
HY-ALL	$w+B$	5.1	83.9	86.9	85.4
	$d+B$	3.3	83.8	91.1	**87.2**
	$d+E$	4.4	83.6	88.3	85.9
	$d+$ Ada-E	3.8	84.3	89.8	87.0
	$d+$ KE	1.9	81.0	**94.3**	87.1

Table 5: Efficiency of different neural network models on vulnerability detection.

Pattern	Training code gadgets	Detection code gadgets	Training time (s)	Detection time (s)
$w+B$	48744	12894	36372.2	156.2
$d+B$	49310	12328	1543.5	3.0
$d+E$	49310	12328	**128.7**	**2.7**
$d+$ Ada-E	49310	12328	2914.0	7.8
$d+$ KE	49310	12328	335.4	11.0

results in a higher computation complexity for Bi-LSTM. Moreover, compared with configuration $d+B$, configuration $d+E$ further reduces the time cost of the training and detection to a few minutes. This can be explained by the fact that the noniterative training mechanism of ELM reduces the computation of parameters. Ada-E improves the precision of ELM by adding the iteration mechanism and introducing the weight mechanism to ELM, but these operations increase the computational complexity. Therefore, the training and detection time of ELM will be multiplied accordingly. KE shows a lower efficiency than conventional ELM because it maps the input and output to a higher dimension for calculation which will result in a larger computational complexity than the former.

Thus, we can conclude that configuration with conventional Bi-LSTM achieves a higher precision, while the configuration with conventional ELM is more effective. Using AdaBoost and kernel function can effectively further improve the precision of conventional ELM in vulnerability detection. In particular, the kernel function achieves a very good precision improvement effect while maintaining higher efficiency than conventional Bi-LSTM.

4.2.2. Results for Q2. To answer the second question, we evaluate the effectiveness of the two vector representation methods, namely, doc2vec and word2vec. We implement experiments with four samples shown in Figure 6. Sample 2 and Sample 4 are labeled as “vulnerable,” while Sample 1 and Sample 3 are not. We collect these four samples from dataset

```
data = ( char*)malloc(100 * sizeof( char));
goodG2BSource(data);
void goodG2B Source(char * &data)
memset(data, "A" , 50 – 1);
data[50 – 1] = "\0" ;
char dest[50] = "";
strcpy(dest, data);
```

(a)

```
char * data;
data = (char*)malloc(100 * sizeof(char));
if(5 == 5)
memset(data, "A" , 100 – 1);
data[100 – 1] = "\0";
char dest[50] = "";
strcpy(dest, data);
```

(b)

```
data = –1;
char inputBuffer[CHAR_ARRAY _SIZE] = "";
if(fgets(inputBuffer, CHAR_ARRAY _SIZE, stdin)! = NULL)
```

(c)

```
char inputBuffer[CHAR_ARRAY _SIZE] = "";
if(fgets(inputBuffer, CHAR_ARRAY _SIZE, stdin)! = NULL)
data = atoi(inputBuffer);
```

(d)

Figure 6: Four samples used to evaluate the effectiveness of the two vector representation methods. (a) Sample 1. (b) Sample 2. (c) Sample 3. (d) Sample 4.

BE-ALL and dataset HY-ALL for two experiments. And both samples are preprocessed with symbolization group of $F + V$.

We evaluate the effectiveness of vector representations by using the similarity measure cosine. The output vector dimension of word2vec is set to 2500, where the output vector dimension of one word is set to 50, and the number of words to represent a paragraph is set to 50. The output vector dimension of doc2vec is set to 250. The reason of making different output vector dimension settings of word2vec and doc2vec is due to the fact that if both dimensions are set to be the same (e.g., 250), then word2vec outputs vector dimension of one word will be 5, or the number of words to represent a paragraph will be 5, which may have a great influence on the effectiveness of vector representation. As a result, the comparison of the effectiveness of word2vec and doc2vec is carried out under the condition that they both use a proper dimension of output vector representation.

From the perspective of vulnerability detection, in terms of the fact that two similar samples are given different labels, it is better to make the similarity between the two vectors after vectorization be small as far as possible, which is conducive to the training of vulnerability detection model using neural network. From Table 6, we find that, for Sample 1 and Sample 2, word2vec outputs vectors with a higher cosine value than doc2vec, while for Sample 3 and Sample 4, it outputs a lower cosine value than doc2vec. Generally, compared with word2vec, doc2vec can output nearly similar or better vector representation with smaller dimension. It can be explained by the fact that, as noted in [16], in order to obtain a fixed length of vector representation, vectors generated by word2vec should be padded with zeros, which may cause the loss of semantic information of the samples. Moreover, it is obvious that a neural network model with low-dimension input vectors can result in good efficiency. We can also observe that, for the same sample in different dataset, doc2vec can output more similar cosine results than word2vec; the biggest output cosine deviation of doc2vec is 0.008, while word2vec results in a value of 0.032. It shows that doc2vec can perform well on large datasets. This conclusion also can be justified by the results in Table 4, where the configurations with doc2vec show better results than the ones with word2vec on HY-ALL.

4.2.3. Results for Q3.

4.2.3. Results for Q3. To answer the third question, we take the configuration $d + B$ and $d + \mathrm{KE}$ as baselines to discuss whether symbolization can further improve the precision of the neural network model. We implement experiments with all the three datasets. And for each dataset, we apply symbolization level from 1 to 3 for preprocessing the datasets.

Table 7 summarizes results of how differently symbolization levels affect the precision of Bi-LSTM. From the perspective of different datasets, symbolization levels have a bigger impact on the precision of Bi-LSTM vulnerability detection model with smaller datasets, which shows a maximum deviation of precision at 3.1% in BE-ALL and 2.6% in RM-ALL. However, with the largest dataset HY-ALL, the maximum precision deviation is 0.9%. This may be because the scale of datasets can affect generalization performance of detection model, while the impact of symbolization is gradually reduced according to the scale becoming smaller. From the perspective of symbolization levels, configuration $d + B$ with the symbolization level 1 shows a better and more stable performance than other symbolization levels, while the symbolization level 2 results in an unstable performance, and the symbolization level 3 shows the worst performance. The main reason is that a high level of symbolization may lose some key vulnerability information in the source codes. Moreover, it should be mentioned that symbolization groups of $F + T$ outperform than symbolization groups of $F + V$ with all datasets; it may be due to the fact that there are many codes related to data type in the source codes; symbolizing them can better capture the vulnerability information.

Table 8 summarizes results of how differently symbolization levels affect the precision of KELM. From the perspective of different datasets, symbolization levels have a big

Table 6: Effectiveness of word2vec compared with doc2vec.

Sample	Tool	Vector dimension	Cosine (BE-ALL)	Cosine (HY-ALL)
1 and 2	word2vec	2500	0.832	0.828
	doc2vec	250	**0.642**	**0.639**
3 and 4	word2vec	2500	**0.482**	**0.514**
	doc2vec	250	0.586	0.578

Table 7: Effect of different symbolization levels on Bi-LSTM precision.

Dataset	Symbolization group	FPR (%)	TPR (%)	P (%)	$F1$ (%)
BE-ALL	F	2.9	86.2	**91.2**	**88.6**
	$F+V$	3.9	83.1	88.1	85.5
	$F+T$	3.2	84.6	90.4	87.4
	$F+\text{V}+T$	3.5	84.5	89.4	86.9
RM-ALL	F	2.8	92.2	94.1	93.1
	$F+V$	3.8	90.3	91.9	91.1
	$F+T$	2.6	93.5	**94.5**	**94.0**
	$F+V+T$	3.4	90.9	92.7	91.8
HY-ALL	F	3.2	87.8	**92.0**	**89.8**
	$F+V$	3.3	83.8	91.0	87.2
	$F+T$	3.2	85.6	91.7	88.5
	$F+V+T$	3.3	86.0	91.1	88.5

Table 8: Effect of different symbolization levels on KELM precision.

Dataset	Symbolization group	FPR (%)	TPR (%)	P (%)	$F1$ (%)
BE-ALL	F	2.1	83.5	93.4	88.1
	$F+V$	1.8	78.7	**93.8**	85.6
	$F+T$	2.0	83.6	93.6	**88.3**
	$F+V+T$	2.3	79.1	92.3	85.2
RM-ALL	F	1.3	88.0	97.0	**92.3**
	$F+V$	1.1	82.7	**97.4**	89.5
	$F+T$	1.2	87.5	97.3	92.1
	$F+V+T$	1.3	83.1	96.8	89.5
HY-ALL	F	1.1	81.8	96.8	88.7
	$F+V$	1.9	81.0	94.3	87.1
	$F+T$	0.87	82.3	**97.5**	**89.3**
	$F+V+T$	1.8	81.6	94.7	87.7

Table 9: Efficiency of different symbolization levels on preprocessing.

Dataset	Symbolization group	Training time (s)
BE-ALL	F	1721.9
	$F+V$	1645.5
	$F+T$	1674.1
	$F+V+T$	1639.3
RM-ALL	F	930.8
	$F+V$	920.8
	$F+T$	908.1
	$F+V+T$	907.4
HY-ALL	F	2762.7
	$F+V$	2693.6
	$F+T$	2692.2
	$F+\text{V}+T$	2665.2

impact on the precision of the KELM-based vulnerability detection model with dataset HY-ALL, which shows a maximum deviation of precision at 3.2%. However, with smaller datasets BE-ALL and RM-ALL, the maximum deviation precision is 1.5% and 0.8%, respectively. This is because the semantic changes of samples generated by different symbolization are smaller in small datasets and larger in large datasets. Therefore, it will cause a large deviation of precision performance. From the perspective of symbolization levels, configuration $d+B$ with the symbolization group of $F+T$ shows the best and most stable performance than other symbolization levels, while the symbolization level 1 results in a better performance than symbolization level 3 with all the datasets. The former phenomenon may be due to the fact that KELM is more suitable for extracting the vulnerability information with dataset preprocessed by symbolization of $F+T$, and the latter one can be explained by the reason mentioned above.

Furthermore, to verify the training efficiency of the proposed multilevel symbol representation, we also give the comparative analysis of time complexity as shown in Table 9. From Table 9, we can observe two phenomena as follows: one is that, for the same dataset, there is a linear downward trend of training time as the symbolization level increases from 1 to 3; the other is that the training time increases correspondingly as the size of dataset increasing. Meanwhile, compared with the symbolization level 1, symbolization level 3 improves training efficiency by about 20% on all three datasets. This can indicate that multilevel symbolization can slightly improve the efficiency of preprocessing, which is not worth mentioning when it is used to improve the precision performance of neural networks.

5. Conclusions

We have made the first effort to use ELM to solve the training efficiency issue of the vulnerability detection model.

Moreover, we then introduce the kernel method to improve the precision of ELM. Experimental results show that ELM with the kernel method is an effective combination of both efficiency and precision. Particularly, for the data preprocessing issue, we find that vector representation using doc2vec performs well on large datasets, and an appropriate symbolization level can effectively improve the precision of vulnerability detection. These experimental conclusions will provide researchers and engineers with guidelines when choosing neural networks and data preprocessing methods for vulnerability detection.

There are several limitations of this paper, which are expected to be researched in the future. First, from more than one kind of single-layer feedforward neural network that could be used for vulnerability detection, we only used ELM in this work. Second, not limited to the kernel method, we expect to explore other methods to improve the precision of ELM subsequently. Third, the datasets used in our experiment are provided by a single source, and more datasets from different sources can be expanded to verify the effectiveness of our proposed approach.

Data Availability

Previously reported vulnerability data were used to support this study and are available at https://github.com/CGCL-codes/VulDeePecker. These prior studies (and datasets) are cited at relevant places within the text as references [16].

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was funded by the Nature Science Foundation of China (grant no. 61872104) and Fundamental Research Funds for the Central Universities in China (grant no. 3072020CF0603).

References

[1] G. Tang, L. Meng, H. Wang et al., "A comparative study of neural network techniques for automatic software vulnerability detection," in *Proceedings of the 2020 International Symposium on Theoretical Aspects of Software Engineering (TASE)*, Hangzhou, China, 2020.

[2] Common vulnerabilities and exposures. https://cve.mitre.org.

[3] National vulnerability database. https://nvd.nist.gov/.

[4] T. Ji, Y. Wu, C. Wang et al., "The coming era of alphahacking?: a survey of automatic software vulnerability detection, exploitation and patching techniques," in *Proceedings of the Third IEEE International Conference on Data Science in Cyberspace*, DSC, Guangzhou, China, 2018.

[5] FlawFinder. http://www.dwheeler.com/flawfinder.

[6] Rough audit tool. https://code.google.com/archive/p/rough-auditing-tool-for-security/.

[7] J. Viega and J. T. Bloch, "A static vulnerability scanner for C and C++ code," in *Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC 2000*, vol. 257, New Orleans, LA, USA, 2000.

[8] Checkmarx, https://www.checkmarx.com/.

[9] Coverity. https://scan.coverity.com/.

[10] H. P. Fortify. https://www.hpfod.com/.

[11] S. Kim, S. Woo, H. Lee, and H. Oh, "VUDDY: a scalable approach for vulnerable code clone discovery," in *Proceedings of the 2017 IEEE Symposium on Security and Privacy*, San Jose, CA, USA, 2017.

[12] Z. Li, D. Zou, S. Xu et al., "Vulpecker: an automated vulnerability detection system based on code similarity analysis," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, Los Angeles, CA, USA, 2016.

[13] S. Ma, F. Thung, D. Lo et al., "Vurle: automatic vulnerability detection and repair by learning from examples," in *Proceedings of the Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security*, Oslo, Norway, 2017.

[14] J. A. Harer, L. Y. Kim, R. L. Russell et al., "Automated software vulnerability detection with machine learning," *CoRR*, 2018.

[15] R. L. Russell, L. Y. Kim, L. H. Hamilton et al., "Automated vulnerability detection in source code using deep representation learning," in *Proceedings of the 17th IEEE International Conference on Machine Learning and Applications*, ICMLA 2018, Orlando, FL, USA, 2018.

[16] Z. LiD. Zou et al., "Vuldeepecker: a deep learning-based system for vulnerability detection," in *Proceedings of the 25th Annual Network and Distributed System Security Symposium, NDSS 2018*, San Diego, CA, USA, 2018.

[17] Z. Li, D. Zou, J. Tang, Z. Zhang, M. Sun, and H. Jin, "A comparative study of deep learning-based vulnerability detection system," *IEEE Access*, vol. 7, pp. 103184–103197, 2019.

[18] K. Cho, Bart van Merrienboer, and D. Bahdanau, "On the properties of neural machine translation: encoder-decoder approaches," in *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, Doha, Qatar, 2014.

[19] G.-B. Huang, Q.-Yu Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proceedings of the 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, vol. 2, pp. 985–990, Budapest, Hungary, 2004.

[20] Word2vec. https://nvd.nist.gov/.

[21] Doc2vec. https://radimrehurek.com/gensim/models/doc2vec.html.

[22] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[23] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[24] B. Chernis, M. Rakesh, and Verma, "Machine learning methods for software vulnerability detection," in *Proceedings of the Fourth ACM International Workshop on Security and Privacy Analytics*, Tempe, AZ, USA, 2018.

[25] Z. Li, D. Zou, S. Xu et al., "Sysevr: a framework for using deep learning to detect software vulnerabilities," *CoRR*, 2018.

[26] Z. Li, D. Zou, S. Xu, and Z. Chen, "Vuldeelocator: a deep learning-based fine-grained vulnerability detector," *CoRR*, 2020.

[27] F. Yamaguchi, N. Golde, and K. Rieck, "Modeling and discovering vulnerabilities with code property graphs," in *Proceedings of the 2014 IEEE Symposium on Security and Privacy, SP 2014*, Berkeley, CA, USA, 2014.

[28] G. Grieco, L. C. Uzal, S. Rawat, J. Feist, and L. Mounier, "Toward large-scale vulnerability discovery using machine learning," in *Proceedings of the Sixth ACM on Conference on Data and Application Security and Privacy, CODASPY 2016*,

E. Bertino, R. S. Sandhu, and P. Alexander, Eds., ACM, New Orleans, LA, USA, 2016.

[29] A. Savchenko, O. Fokin, A. Chernousov, O. Sinelnikova, and S. Osadchyi, "Deedp: vulnerability detection and patching based on deep learning," *Theoretical and Applied Cybersecurity*, vol. 2, p. 8, 2020.

[30] Q.-Q. Tao, S. Zhan, X.-H. Li, and T. Kurihara, "Robust face detection using local cnn and svm based on kernel combination," *Neurocomputing*, vol. 211, pp. 98–105, 2016.

[31] P. Liang, W. Li, D. Liu, and J. Hu, "Large-scale image classification using fast svm with deep quasi-linear kernel," in *Proceedings of the 2017 International Joint Conference on Neural Networks*, pp. 1064–1071, IJCNN), Anchorage, AL, USA, 2017.

[32] W. Zhang, M. Xia, and J. Zhu, "An efficient hierarchical identification method with kernel-based svm for equivalent systems of aircrafts," *IEEE Access*, vol. 7, pp. 83243–83250, 2019.

[33] Li Lu, C. Wang, W. Li, and J. Chen, "Hyperspectral image classification by adaboost weighted composite kernel extreme learning machines," *Neurocomputing*, vol. 275, pp. 1725–1733, 2018.

[34] L. Wolf, H. Yair, and N. Dershowitz, "Joint word2vec networks for bilingual semantic representations," *International Journal of. Computational Linguistics and Applications*, vol. 5, no. 1, pp. 27–42, 2014.

[35] V. Quoc, "Le and Tomas Mikolov. Distributed representations of sentences and documents," in *Proceedings of the 31th International Conference on Machine Learning, ICML 2014*, Beijing, China, 2014.

[36] H. Yuan, B. Wang, and L. Niu, "Kernel extreme learning machine for learning from label proportions," *Lecture Notes in Computer Science*, vol. 400, p. 409, 2018.

[37] G. B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.

[38] M. Pendleton, R. Garcia-Lebron, J.-H. Cho, and S. Xu, "A survey on systems security metrics," *ACM Computational Survey*, vol. 49, no. 4, p. 62, 2017.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 9994127, 13 pages
https://doi.org/10.1155/2021/9994127

Research Article

Malicious URL Detection Based on Improved Multilayer Recurrent Convolutional Neural Network Model

Zuguo Chen, Yanglong Liu, Chaoyang Chen, Ming Lu, and Xuzhuo Zhang

School of Information and Electrical Engineering, Hunan University of Science and Technology, Xiangtan 411201, Hunan, China

Correspondence should be addressed to Yanglong Liu; 1804060104@mail.hnust.edu.cn

Received 13 March 2021; Accepted 14 May 2021; Published 27 May 2021

Academic Editor: Entao Luo

The traditional malicious uniform resource locator (URL) detection method excessively relies on the matching rules formulated by the network security personnel, which is hard to fully express the text information of the URL. Thus, an improved multilayer recurrent convolutional neural network model based on the YOLO algorithm is proposed to detect malicious URL in this paper. First, single characters are mapped to dense vectors using word embedding, and the dense vectors are participated in the training process of the whole model according to the structural characteristics of the URL in the method. Then, the CSPDarknet neural network model based on the improved YOLO algorithm is proposed to extract features of the URL. Finally, the extracted features are used to evaluate malicious URL by the bidirectional LSTM recurrent neural network algorithm. In order to verify the validity of the algorithm, a total of 200,000 URLs are collected, including 100,000 normal URLs labeled "good" and 100,000 malicious URLs labeled "bad". The experimental results show that the method detects malicious URLs more quickly and effectively and has high accuracy, high recall rate, and high accuracy compared with Text-RCNN, BRNN, and other models.

1. Introduction

With the rapid development of Internet technology, network crime is becoming more and more serious, which brings heavy losses for personal network privacy and property security [1]. However, mixing well-known URLs with malicious URLs to cause user confusion and achieve intrusion attacks on the host is one of the most common attack methods. At present, malicious URLs are detected by using rule matching and the black-and-white list [2, 3]. But these methods are excessively dependent on the knowledge breadth of security personnel, which increases the possibility of false blocking of malicious URLs. Moreover, when these detection models are used to detect fishing URL of unknown attack types, there will be a great probability of false blocking or missed blocking.

To solve these problems, scholars at home and abroad have done a lot of research studies. Anwar et al. [2] and Li et al. [4] proposed a method combining linear and nonlinear spatial transformation for URL recognition and detection. The method significantly improves the accuracy of URL recognition and detection using a support vector machine and neural network. Vu et al. [5] proposed a new cost-sensitive classifier to detect malicious URLs in large enterprise networks. The method classifies the input URL into benign, unknown, and malicious and uses the cost matrix to select the most relevant features and to control the model misclassification. It can effectively reduce the false detection rate of the malicious URL. Yang et al. [6] proposed a URL feature representation method based on malicious keywords. The method uses a convolutional gated recurrent unit (GRU) neural network to replace the feature collection of the original pooling layer in the time dimension, which obtains a high-precision result. Yuan et al. [7] proposed a parallel neural joint model algorithm for analyzing and detecting malicious URL. The semantic features and text features are combined by merging parallel joint neural network and independent recurrent neural network in the algorithm, which can improve the detection accuracy of the fishing URL of unknown attack types. Yang et al. [8] proposed a multidimensional feature detection method for malicious URLs based on deep reinforcement learning. The method first extracts the sequence features of a given URL, classifies them quickly through deep learning, and fuses the statistical features, the web page code features, and the web page text features into multidimensional features for detection, which can obtain higher detection accuracy of the malicious URL. Wang et al. [9] proposed a bidirectional LSTM algorithm based on convolutional neural network and independent recurrent neural network. The algorithm extracts the feature information of malicious URL binary file and uses the Word2Vec algorithm to train URL word vector characteristics and extract URL static vocabulary features, which can improve the detection accuracy of the malicious URLs. Chen et al. [10] proposed a multifeature information fusion-based identification algorithm in a complex environment, which achieves a better effect.

These methods are difficult to find the appropriate vector space to represent a single character in the process of URL numerical expression because of the randomness of the composition characters of URL strings, which results in low recognition accuracy of the malicious URL. A malicious URL detection model based on the combination of multilayer convolutional neural network and bidirectional recurrent neural network is proposed in the paper. First, the separated characters and the special characters are filtered by the model according to the structural characteristics of the URL. Then, the model unifies the lengths of all URL characters, intercepting the longer URL and filling the shorter URL with zero. By participating in the training of the neural network model through the word embedding layer, each character in the URL can be mapped into a dense vector in the embedding space, so each URL can be represented as a two-dimensional tensor. Next, the improved continuous multilayer convolutional neural network, which is based on the CSPDarknet neural network in the YOLO model, is used for feature extraction. The convolution layer in the network uses a one-dimensional convolutional neural network to extract the local context in the sequence. Finally, the results of feature extraction are input into the bidirectional recurrent neural network, and the network detects malicious URLs in positive and negative directions.

The main contributions of the paper are summarized as follows:

(1) CSPDarknet network model of the YOLO algorithm is improved by the one-dimensional convolutional neural network, which is used for feature extraction of URL sequence

(2) bidirectional recurrent neural network is used to process the URL sequence after feature extraction

(3) The translation invariance of the one-dimensional convolutional neural network is combined with the sequence sensitivity of RNN

The remainder of the paper is organized as follows. Section 2 introduces the character statistics and encoding of URL. Section 3 introduces how to combine convolutional neural networks and recurrent neural networks and proposes an improved multilayer convolutional recurrent neural network model based on the YOLO algorithm. Section 4 carries out simulation experiments and data analysis and verifies the advancement of this method through comparative experiments. Section 5 summarizes the conclusions.

2. URL Data Preprocessing

2.1. URL String Preprocessing. The research in this article mainly uses the Windows system, and it is unnecessary to match a case for any URL, so the uppercase letters in the URL can be transformed into lowercase. The smallest granularity, character, is selected as the smallest processing unit. Based on the statistics of the frequency of various characters in a large number of positive and negative datasets, this article deletes the low-frequency special characters to ensure that each URL provides the most useful information as much as possible while reducing the complexity of the URLs. This paper collects more than 400,000 URLs and counts the frequency of characters in each URL. The results are shown in Figure 1. The abscissa represents the index of the character, and the ordinate represents the frequency of character occurrences. It can be seen from Figure 1 that the occurrence frequency of characters after the 45th index is very low, so the characters after the 45th index can be deleted from the URL, and the influence on the feature information in the URL can be ignored. At this point, the length of each URL is mostly inconsistent, so the length of each URL needs to be standardized. This article counts the length of each URL in the dataset and finds that the average length is 48 characters. Therefore, all URLs in the dataset are uniformly processed into 48 characters in length. The long part is truncated, and the short part is filled with zero to ensure that each URL has the same length.

2.2. Character Encoding. URL can be regarded as a series of text sequences, but as most machine learning algorithms, the deep learning model cannot directly receive the original text sequence as input, and it can only process numerical tensor. Therefore, how to encode the information contained in the URL as a numerical expression is an important prerequisite for model recognition and detection. At present, the common coding methods are based on N-gram [11], one-hot [12], and word embedding methods [13]. N-gram is a word segmentation method that does not preserve text order. It is often processed in shallow natural language, while URL detection depends on text order. One-hot encoding usually maps the text sequence to a high-dimensional sparse tensor. This method cannot calculate the similarity between tensors, and it is easy to fall into high-dimensional disaster in the actual neural network training. However, there is a certain similarity among malicious URLs in many cases. In order to solve these problems, this paper uses the word embedding method, which can integrate more information into lower dimensions. This paper compares the one-hot word vector with the word embedding vector, as shown in Figure 2. In Figure 2, each row of squares represents a character vector, and each square with different colors represents different values. For instance, the one-hot word vector associates each character with a unique integer index i and converts this integer index into a binary character vector with length M.

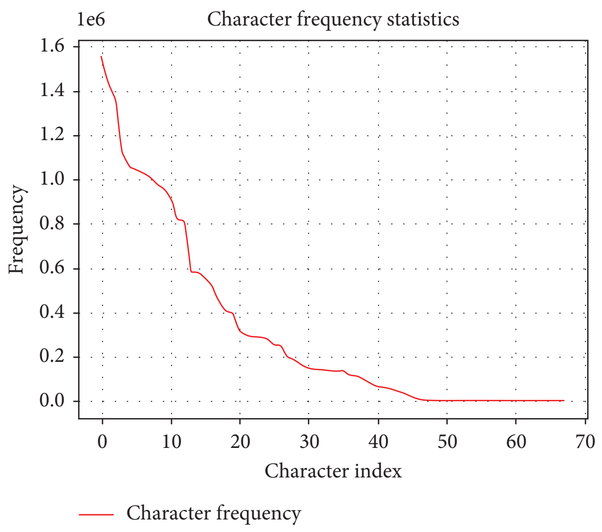


Figure 1: Character frequency statistics.

This character vector only has the value of the i-th index of 1, and the rest are 0. In the word embedding vector, each square represents different values. Therefore, the word embedding vector can fully express more information in lower dimensions than the one-hot word vector.

In order to obtain the embedding space model that maps characters to dense vectors, this paper uses the embedding layer to learn word embedding and to participate in the training process of the entire neural network model. In the process of model training, the weight parameters of the embedding layer are adjusted through the reverse transmission of the entire network. With the gradual convergence of the whole network model, the embedding space model of character mapping to vector will also tend to be stable, so that the obtained embedding space structure can facilitate the use of downstream neural network models.

3. Malicious URL Detection Model Based on Improved Multilayer Recurrent Convolutional Neural Network

3.1. Convolutional Neural Network. In recent years, the neural network has achieved a major breakthrough in the field of machine vision. One of the important reasons is the emergence of the convolutional neural network, which enables the neural network to perform convolution operation on images and extract information features from part of images [14]. At the same time, one-dimensional convolutional neural networks also perform extremely well in sequence processing, such as speech recognition and machine translation. This paper will realize malicious URL detection, which also depends on the character sequence. Therefore, this paper uses one-dimensional convolutional neural network to process the character vector of URL so as to achieve feature extraction. The working principle of one-dimensional convolutional neural networks is shown in Figure 3.

Figure 3 is a two-dimensional numerical tensor formed by character-level vectorization of a single URL. The window sliding on this tensor will extract the surrounding feature blocks at the location, and then each block does a tensor product with the same weight matrix (or called convolution kernel). Reusing multiple different convolution kernels will form multiple sets of vectors, and the one-dimensional convolution operation is completed by spatial reorganization of all vectors.

3.2. Bidirectional Recurrent Neural Network. A recurrent neutral network (RNN) has an additional information memory function in its hidden layer compared with the full connection layer. The input of the hidden layer at each time step includes not only the input of the current time step but also the output of the previous time step hidden layer [15]. This is conducive to information interaction between neural units in the same layer and realizes the memory function of past information. The method of RNN processing sequence is to traverse every element of all sequences and save a state, respectively. The output of the current event step is used as the state input of the next time step to form a neural network with an internal loop. The RNN-specific network structure is shown in Figure 4.

In Figure 4, X_t represents the input layer at the t-th time step, O_t represents the output value at the t-th time step, and S_t represents the state value at the t-th time step. U, W, and V represent weight matrices. The output value O_t is calculated in equation (1), and the state value S_t is calculated in equation (2).

$$O_t = g(VS_t), \tag{1}$$

$$S_t = f(UX_t + WS_{t-1}), \tag{2}$$

where g represents the activation function of neurons in the output layer and f represents the activation function of neurons in the hidden layer.

Theoretically, RNN can remember all the information that it traversed many time steps before. But practically, it is impossible to learn such long-term dependence because of the gradient disappearance problem. Therefore, this paper uses long short-term memory (LSTM) to build the neural network model. In essence, LSTM is a variant network of RNN. It adds a method to carry information across multiple time steps. To be specific, it allows information traversed by past time steps to reenter the network at future time steps, so as to solve the problem of gradient disappearance [16]. The unit structure of the LSTM neural network is shown in Figure 5, which consists of forgetting gate, LSTM unit state, input gate, and output gate.

In Figure 5, x_t represents the input vector at time t, h_t represents the hidden state at time t, and c_t represents the LSTM unit state at time t. The forgetting gate receives the hidden state h_{t-1} at the previous time and the input vector x_t at the current time and transmits them to the Sigmoid function. The range of the output value f_t is [0, 1]. If the output value is close to 0, it means information forgotten, and if it is close to 1, it means information retention. Therefore, the forgetting gate determines the abandonment

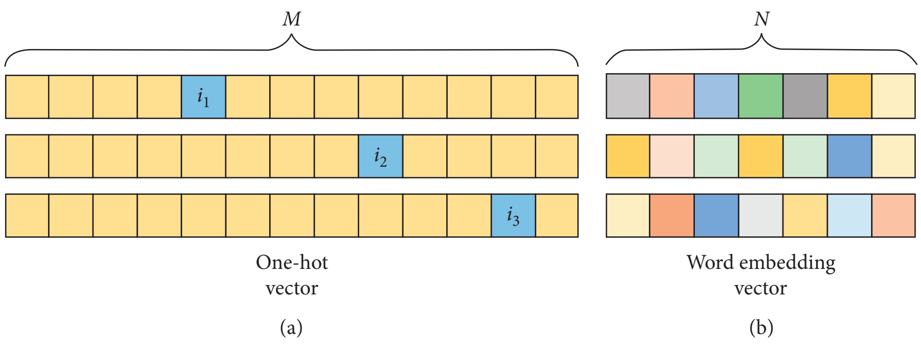


Figure 2: Comparison between one-hot word vector and word embedding vector.

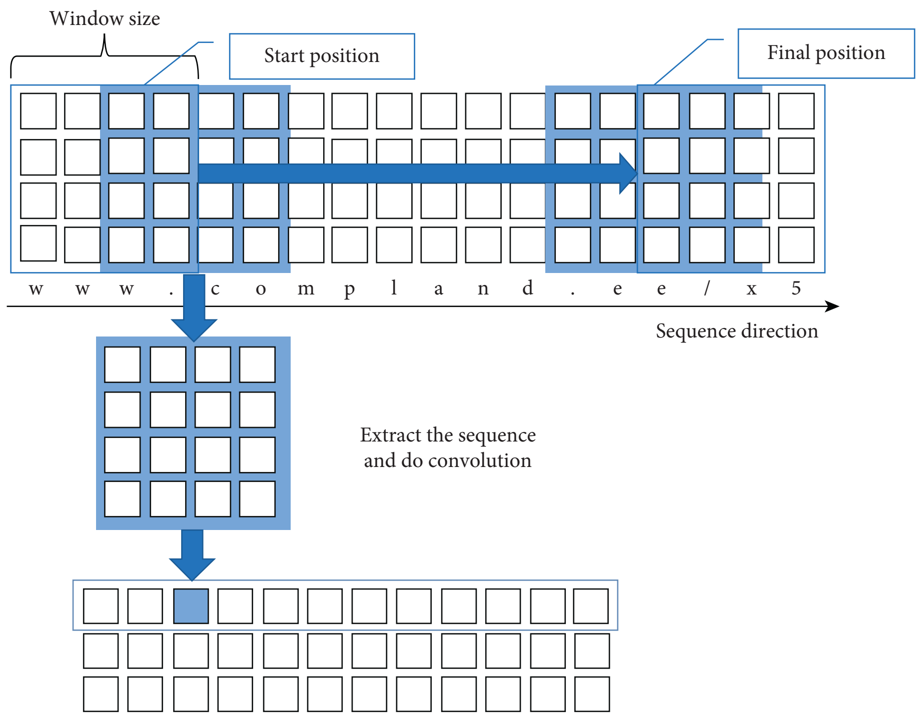

Window size
Start position
Final position
w w w . c o m p l a n d . e e / x 5
Sequence direction
Extract the sequence and do convolution

Figure 3: One-dimensional convolution operation principle diagram.

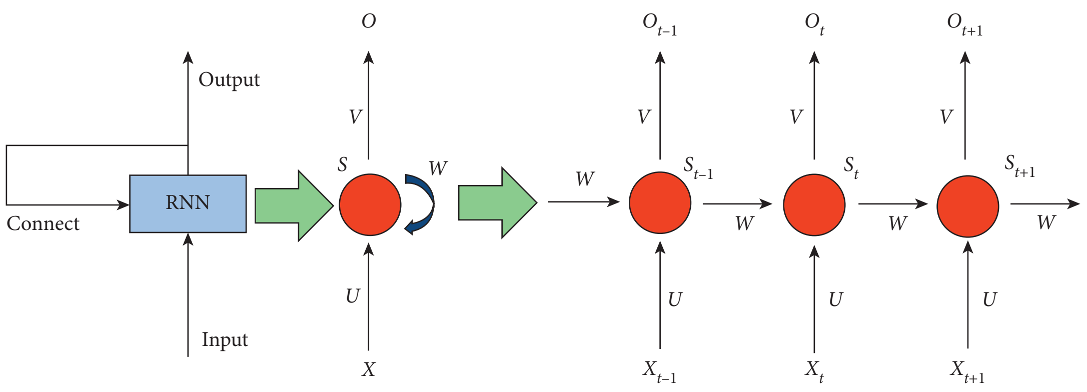


Figure 4: The structure of the RNN.

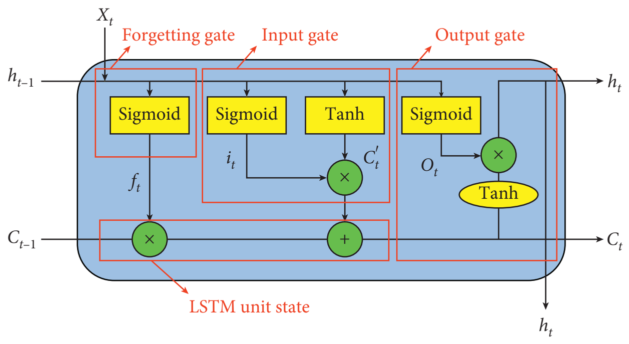


FIGURE 5: LSTM neural network unit.

and retention of information. The calculation process of the output value f_t of the forgetting gate is as follows:

$$f_t = \text{Sigmoid}\left(W_f \cdot [h_{t-1}, x_t] + b_f\right), \quad (3)$$

where W_f and b_f, respectively, represent the weight and bias in the forgetting gate.

The input gate receives the hidden state h_{t-1} at the previous moment and the input vector x_t at the current moment. They are transmitted to the Sigmoid function and the Tanh function simultaneously. The range of output value i_t of the Sigmoid function is [0, 1]. The closer the output value is to 0, the less important the information is, and the closer the output value is to 1, the more important the information is. The range of output value c_t' of the Tanh function is [–1, 1], which is used to output a new candidate vector. Then, the output values of the Sigmoid function and the Tanh function are multiplied, so that the output value of the Sigmoid function can determine which information is important in the candidate vector output by the Tanh function and can be saved. The calculation processes of the Sigmoid function output value i_t and the Tanh function output value c_t' are shown in equations (4) and (5), respectively:

$$i_t = \text{Sigmoid}\left(W_i \cdot [h_{t-1}, x_t] + b_i\right), \quad (4)$$

$$c_t' = \text{Tan}\,h\left(W_c \cdot [h_{t-1}, x_t] + b_c\right), \quad (5)$$

where W_i and b_i, respectively, represent the weight and bias of the Sigmoid function in the input gate and W_c and b_c represent the weight and bias of the Tanh function in the input gate, respectively.

The LSTM unit state receives the unit state c_{t-1} at the previous time, multiplies it with the output value f_t of the forgetting gate, and then adds the output value of the input gate to get the cell state c_t at the current time, so as to update the unit state in the LSTM neural network. The calculation formula of c_t is as follows:

$$c_t = f_t \cdot c_{t-1} + i_t \cdot c_t'. \quad (6)$$

The output gate receives the hidden state h_{t-1} at the previous moment and the input vector x_t at the current moment and transfers them to the Sigmoid function. At the same time, the obtained LSTM unit state c_t is transferred to the Tanh function. Then, the output value of the Sigmoid function is multiplied by the output value of the Tanh function to obtain the hidden state h_t at the current moment. The calculation process of h_t is as follows:

$$h_t = \text{Sigmoid}\left(W_s \cdot [h_{t-1}, x_t] + b_s\right) \times \text{Tanh}\left(c_t\right), \quad (7)$$

where W_s and b_s, respectively, represent the weight and bias of the Sigmoid function in the output gate.

Malicious URL detection is strictly dependent on character order, and LSTM recurrent neural networks are processed in a single forward sequence. Therefore, this article, in order to further explore the relationship between the future state and the past state, adopts the bidirectional recurrent neural network, which is processed in a front-to-back and back-to-front direction, respectively [17]. Finally, the processing results of the two are combined to achieve more comprehensive data mining. The implementation structure of bidirectional recurrent neural network is shown in Figure 6, which is mainly composed of the input layer, hidden layer, and output layer.

In Figure 6, the first column represents the input layer of the bidirectional recurrent neural network, and the middle two columns represent the forward hidden state and the reverse hidden state, respectively. Their calculation formulas are shown as follows:

$$F_t = \phi\left(X_t W_{xh}^F + F_{t-1} W_{hh}^F + b_h^F\right), \quad (8)$$

$$B_t = \phi\left(X_t W_{xh}^B + B_{t+1} W_{hh}^B + b_h^B\right), \quad (9)$$

where ϕ represents the activation function of the hidden layer, X_t represents the input at time t, h represents the number of positive and negative hidden units, W_{xh}^F and W_{hh}^F represent positive weights, W_{xh}^B and W_{hh}^B represent negative weights, and b_h^F and b_h^B represent positive bias and negative bias, respectively. Then, the forward hidden state F_t is connected with the reverse hidden state B_t obtained above so as to obtain the hidden state H, and then the hidden state H is input to the output layer O_t. The calculation process is shown as follows:

$$O_t = H_t W_{hq} + b_q, \quad (10)$$

where H_t represents the hidden state at time t, q represents the number of output units, W_{hq} represents the weight from hidden unit to output, and b_q represents the output bias.

3.3. The Establishment of Malicious URL Detection Network Model. Inspired by the YOLO algorithm, this paper applies the CSPDarknet neural network model to extract the feature of character vectors. The YOLO algorithm is used for image target detection. It realizes high-precision and high-efficiency real-time detection. This article is based on the YOLOv4 algorithm to realize the malicious URL detection of the multilayer convolutional recurrent neural network model. The YOLOv4 algorithm is mainly composed of three network components: Backbone, Neck, and Head [18]. The Neck network component is mainly used to generate feature pyramids in image target detection and identify targets of

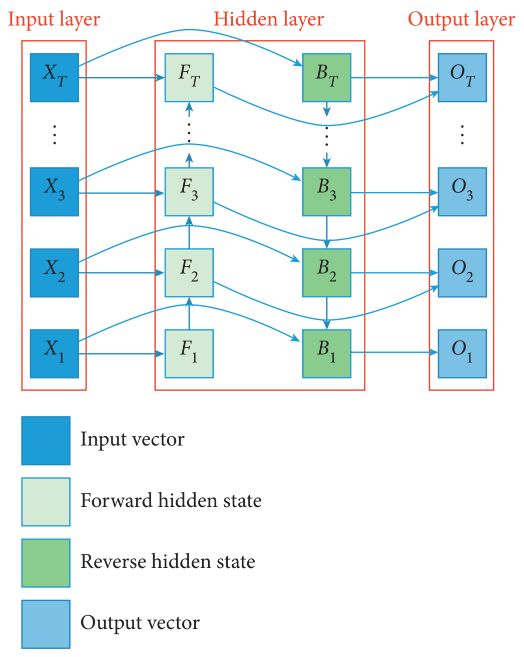


Figure 6: Structure diagram of bidirectional recurrent neural network.

different sizes by detecting zoom tensors of different scales. The Head network component is mainly used for the final anchoring of image target detection while generating target category probability, target score, and the position vector of the bounding box. CSPDarknet is mainly used to extract rich feature information from the image. It integrates the feature information into the feature map from top to bottom and gradually reduces the size. When using the CSPDarknet network to extract features of the URL numerical tensor, it can downsample the high-dimensional URL tensor to a low-dimensional space, which improves the detection speed of the model. Moreover, this paper uses a one-dimensional convolutional neural network to process sequence tensor in the CSPDarknet network. When the CSPDarknet network learns a certain local feature on the URL sequence, because a one-dimensional convolutional neural network has translation invariance, it can identify this local feature at any position on any URL. When dealing with each URL, this paper first uses word embedding to carry out numerical vectorization with characters as the smallest unit to obtain a two-dimensional tensor. Then, the CSPDarknet network can be improved and applied to the preprocessing step of recurrent neural network processing sequence front-end, so as to integrate the information after feature extraction of the convolutional neural network, and the bidirectional LSTM is used to identify malicious URLs. The whole network structure is shown in Figure 7.

In Figure 7, the model first receives the URL through word embedding processing for character-level vectorization to form a two-dimensional digital tensor. Each square in the tensor represents a character vector. The two dimensions of this two-dimensional tensor, respectively, represent the character length of the URL and the space vector of each character. However, when processing multiple URLs, a dimension is added to represent the number of URLs. Therefore, word embedding can convert the URLs into a three-dimensional tensor. Then, the CSPDarknet network framework, which is mainly composed of CBM, ResUnit, and CSPn, is used to extract the features of this three-dimensional tensor. The CBM component is composed of a one-dimensional convolutional neural network, batch normalization, and Mish activation function. The one-dimensional convolutional neural network can process the three-dimensional tensor obtained by the URL after word embedding. Moreover, the size of the convolution kernel of this one-dimensional convolutional neural network is 3, and the stride is 1. Batch standardization is to standardize not only the input layer but also the input (before activation function) of each intermediate layer. It is conducive to gradient propagation [19]. The Mish activation function is shown in equation (11), and its shape is similar to the ReLU activation function. However, the Mish activation function also allows relatively small negative gradient inflow in the case of negative values. The Mish activation function ensures that the positive value is unbounded and avoids the phenomenon of saturation.

$$\operatorname{Mish}(x) = x \cdot \operatorname{Tan} h\left(\ln\left(1 + e^{x}\right)\right). \tag{11}$$

Network component ResUnit indicates residual connection after two CBM operations. The residual connection solves the problem of gradient disappearance [20]. Its principle is to take the output of the previous layer as the input of the latter layer, so as to create a shortcut to let information directly enter the deep network, which effectively avoids the problems of gradient disappearance and gradient explosion. The calculation of CSPn components has two processing directions. The first processing direction is to

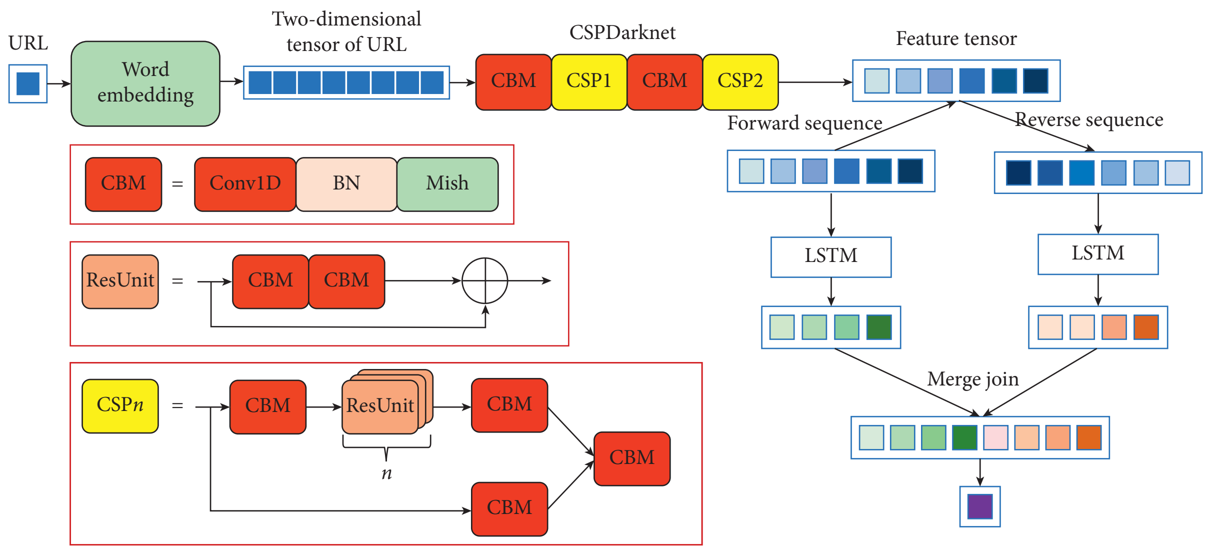


Figure 7: Malicious URL detection network model.

calculate the input value through the CBM network. The calculated results are connected by *n* times residuals, and the connected results are then calculated through the CBM network. The second processing direction is to calculate the input value through the CBM network. But the calculated results will be connected to the results of the first processing direction and output. After extracting feature information by the multilayer convolutional neural network, the obtained feature sequence tensor is processed by a bidirectional LSTM recurrent neural network, and the detection is performed in two directions: front to back and back to front. Then, the two tensors processed in two directions are spliced into a three-dimensional tensor, which is then expanded to a one-dimensional tensor. Finally, the output of the whole model is realized through the dense layer and the Sigmoid activation function. This paper stipulates that the closer the output of the model is to "1," the URL is labeled as "good", the closer the output of the model is to "0," the URL is labeled as "bad".

4. Experiment Results and Analysis

This article collected 200,000 URLs, including 100,000 normal URLs labeled as "good" and 100,000 malicious URLs labeled as "bad". This article randomly selected 90,000 normal URLs and 90,000 malicious URLs as training datasets, and the remaining 10,000 normal URLs and 10,000 malicious URLs are treated as test datasets. Twenty percentage of the training dataset is selected as the validation dataset in the process of model training.

In this paper, we use the embedding layer to learn character embedding and participate in the training process of the whole neural network model. Therefore, each character of URL can learn the unique spatial vector representation in the convergence process of the whole network model. After experiments, in order to analyze the character vector more clearly, the URL character vector is reduced to a three-dimensional space through the PAC algorithm, as shown in Figure 8. Each color in the figure represents a different character. Although the dimensionality is reduced, it can still be seen that the number charactered from "0" to "9" are more concentrated, the alphabetic characters from "a" to "z" are more concentrated, and the division between digital characters and alphabetic characters is more obvious. It can also be concluded from the actual URL that most of the feature information in the URL is represented by letters, and numbers are usually used to represent parameters. Therefore, it can be considered that the character vector obtained by model training has a good effect and provides a good foundation for the following feature extraction and malicious detection.

This article uses 25 URLs for training in small batches per round, with a maximum of 20 rounds. At the same time, in order to avoid the occurrence of overfitting, a callback function is added when training the model, and the checkpoint of the model is set and terminated early. If the target indicators monitored during the training process are no longer improved within the specified 20 rounds, the training can be terminated in advance, and the model weight can be saved. The accuracy and loss values of the model are shown in Figures 9(a) and 9(b), respectively. The red curve represents the accuracy and loss values of the training dataset, and the blue curve represents the accuracy and loss values of the validation dataset. It can be seen from the figures that, with the increase of training iterations, the training accuracy and loss values tend to converge, and the accuracy and loss values of the validation dataset are consistent with the training data. It indicates that the model can effectively identify malicious URLs.

In order to prove the advancement of the malicious URL identification method proposed in this paper, comparative experiments are carried out among the Darknet network model based on YOLOv3, the traditional bidirectional recurrent neural network, the traditional recurrent neural network, the RCNN neural network, and the neural network based on the full connection layer. Figures 10(a) and 10(b)

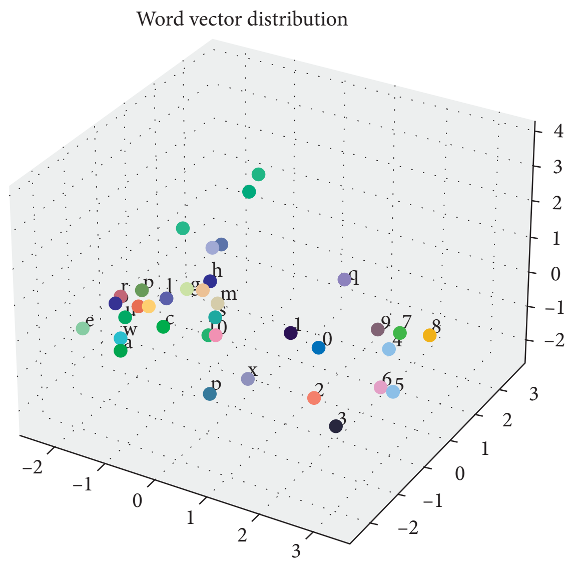


FIGURE 8: Three-dimensional mapping of character space vector.

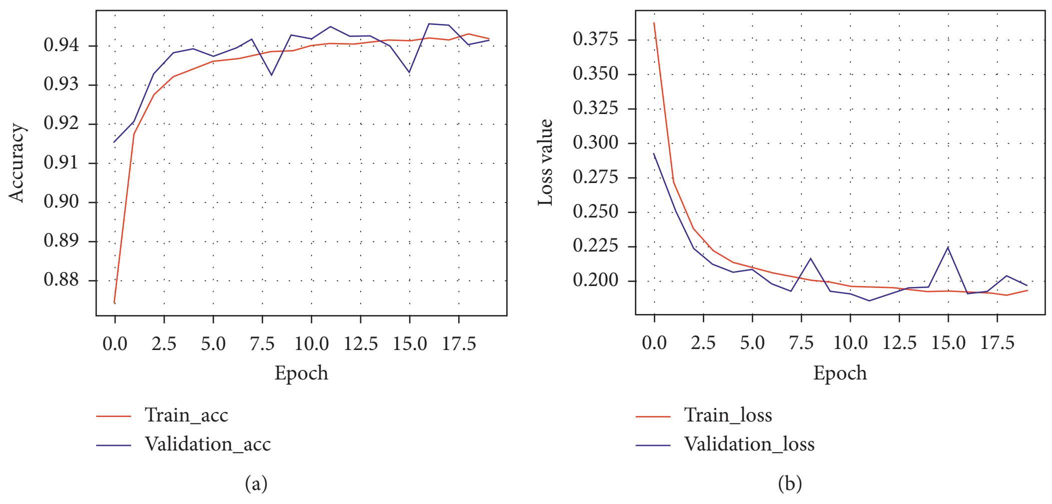


FIGURE 9: Training process of CSPDarknet model: (a) accuracy of model training and (b) loss values of model training.

are the training process of the Darknet network model based on YOLOv3. With the increase of the number of training iterations, the accuracy of the training dataset and the validation dataset are gradually increased. At the same time, the loss value of the training dataset and the validation dataset are gradually reduced. The final accuracy can be stabilized at about 94%, and the loss value can be stabilized at about 0.19. It can be seen that the training results of the Darknet network model based on YOLOv3 and the CSPDarknet network model based on YOLOv4 are similar.

Figures 11(a) and 11(b) are based on the traditional bidirectional recurrent neural network training process. It can be seen that the accuracy of the validation dataset is higher than that of the training dataset at the beginning stage, and the loss value of the validation dataset is lower than that of the training dataset. However, with the increase of iterations, the accuracy of the validation dataset is gradually lower than that of the training dataset, and the loss value of the validation dataset is gradually higher than that of the training dataset. These indicate that overfitting occurred in the later stage of the model training, and the generalization ability of the model is reduced.

Figures 12(a) and 12(b) are neural network model training process based on RCNN. It can be seen from the two figures that the whole model no longer improves in the ninth iteration of training, that is, it tends to be convergent and stable. The accuracy of the validation dataset is stable at about 92%, and the loss value is stable at about 0.22. It can be also concluded that the RCNN model begins to appear overfitting in the fourth iteration.

Figures 13(a) and 13(b) are neural network model training process based on RNN. It can be seen from these two figures that while the accuracy and loss values of the training dataset gradually converge, the accuracy and loss values of the validation dataset also gradually converge, but the model is unstable during the convergence process. It can be concluded from the convergence results that the accuracy

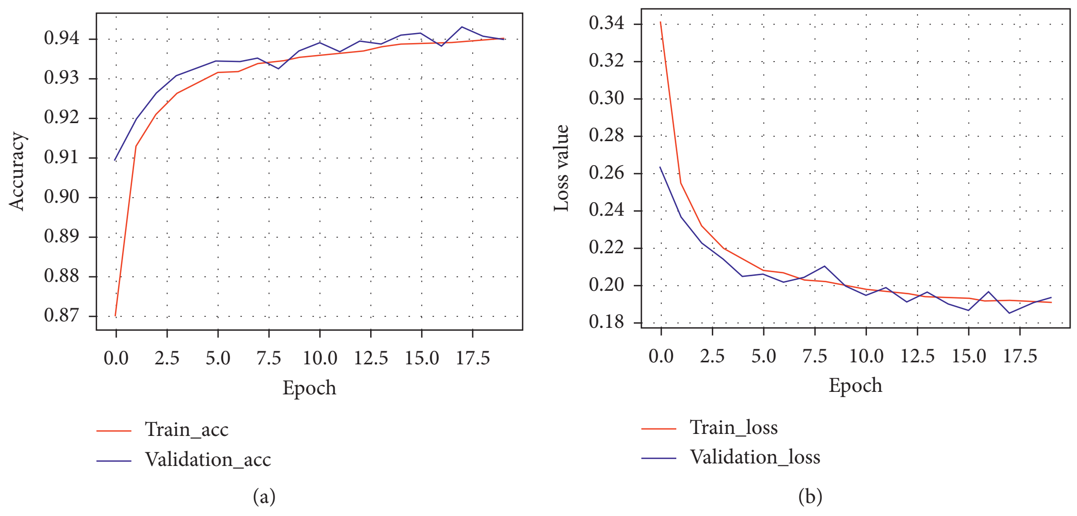


FIGURE 10: Training process of Darknet model: (a) accuracy of model training and (b) loss values of model training.

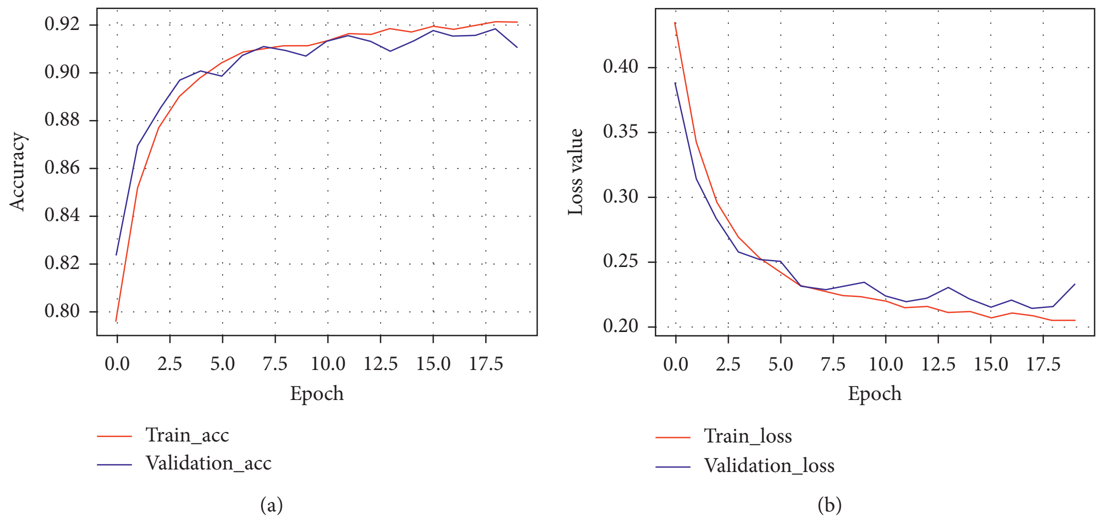


FIGURE 11: Training process of BRNN model: (a) accuracy of model training and (b) loss values of model training.

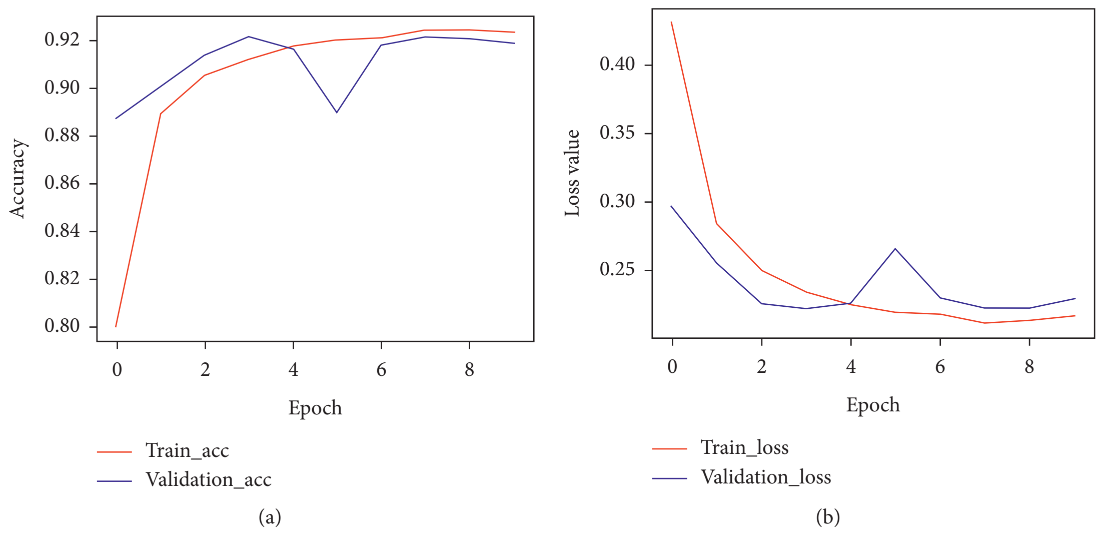
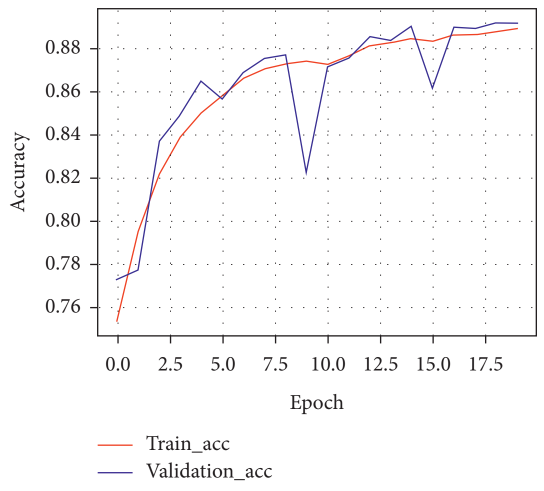
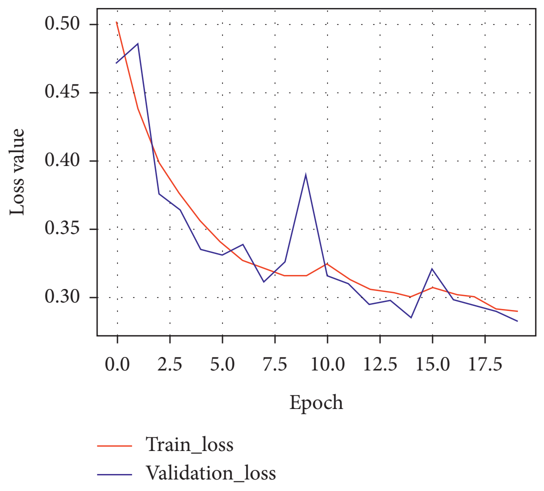


FIGURE 12: Training process of RCNN model: (a) accuracy of model training and (b) loss values of model training.



(a)



(b)

Figure 13: Training process of the RNN model: (a) accuracy of model training and (b) loss values of model training.

of the validation dataset is no more than 90%, and the minimum loss value is no less than 0.32.

Figures 14(a) and 14(b) are neural network models based on fully connected layers. It can be seen from the two figures that the training effect of the model no longer improves at the tenth iteration. In addition, the model is serious overfitting. The accuracy of the validation data set is about 86%, and the loss value is about 0.33. However, it is still at an unstable state. The effect of the model is extremely worse than the other five types of models.

It can be seen from the comparative analysis of Figures 9 and 14 that, in the whole training process, if we take the loss value reduced to 0.2 as the standard, the loss value of the two models, namely, the improved model based on CSPDarknet network in YOLOv4 and the improved model based on Darknet network in YOLOv3, will drop roughly the same. But if we take the accuracy increased to 94% as the standard, the improved model based on CSPDarknet has been completed in the 10th iteration, and the improved model based on Darknet is completed after the 17th iteration. Therefore, the improved model based on CSPDarknet has a faster convergence rate, and the accuracy is slightly higher than the improved model based on Darknet. The traditional bidirectional recurrent neural network model, the RCNN model, and the network model based on the full connection layer exist overfitting phenomena, and the severity of overfitting increases sequentially. Although there is no overfitting phenomenon based on the traditional recurrent neural network model, the recognition accuracy of the model is low, and there is a large fluctuation in the convergence process. If the six models are sorted according to the accuracy of the validation data set, then they are ranked from high to low as follows: the CSPDarknet network model based on YOLOv4, the Darknet network model based on YOLOv3, the traditional bidirectional recurrent neural network, the RCNN neural network, the traditional recurrent neural network, and the neural network based on the full connection layer.

The abovementioned trained models are used in the test dataset. The test dataset contains 10,000 normal URLs and 10,000 malicious URLs. The evaluation results are shown in Figure 15.

Figure 15 shows the performance of the six models in a dataset of 20,000 URLs. Obviously, the improved model based on CSPDarknet and the improved model based on Darknet have higher recognition accuracy and lower loss value. At the same time, since the activation function used in the output layer of the models is sigmoid, the difference in the loss value function is not large, and the accuracy of the improved model based on CSPDarknet is greater than that of the improved model based on Darknet. It is considered that the improved model based on CSPDarknet is slightly better than that based on Darknet. Finally, it can be seen that the accuracy and loss values of the other four models on the test dataset are worse than those of the improved multilayer convolutional recurrent neural network model proposed in this paper.

At the same time, if the RNN model is compared with the network model based on the full connection layer, we can conclude that the detection of malicious URL is dependent on the character sequence, and the relationship between the context characters in the URL can be found based on RNN, thus improving the accuracy of the model. If the RNN model is compared with the BRNN model, we can conclude that the bidirectional recurrent neural network can process the URL sequence in two directions: front to back and back to front. By combining the relationship between the future sequence and the past sequence at the current time step, the accuracy of the model is further improved. If the RCNN model is compared with the BRNN model, we can conclude that when the one-dimensional convolutional neural network is combined with the bidirectional recurrent neural network, the convolutional neural network can first extract the feature information in the URL and then hand it to the bidirectional recurrent neural network, which can effectively improve the recognition accuracy.

In order to better evaluate the superiority of each model in identifying malicious URLs, this paper selects precision,

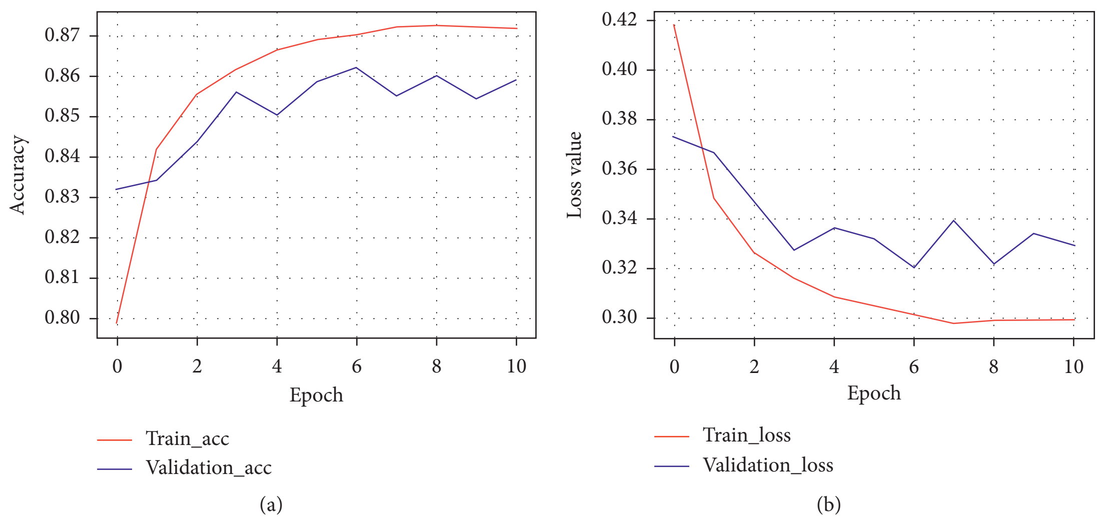


Figure 14: Training process of full connection layer network model: (a) accuracy of model training and (b) loss values of model training.

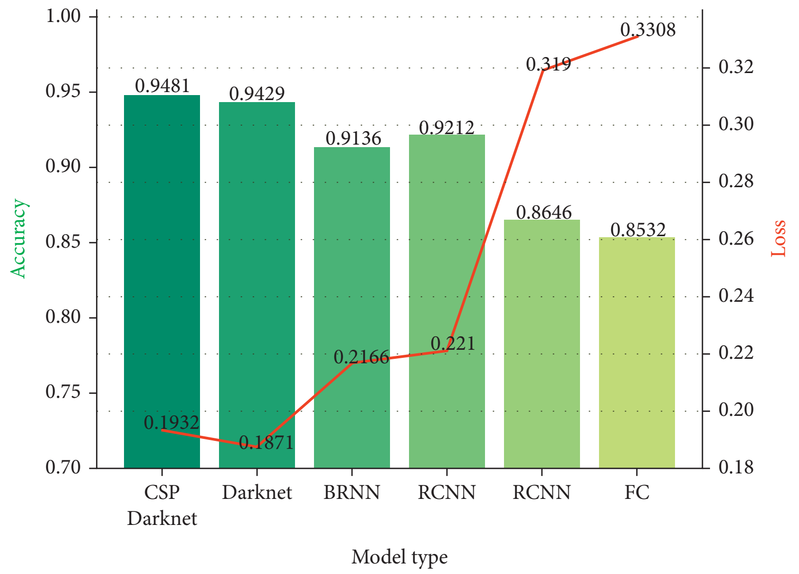


Figure 15: The accuracy and loss of the test set.

recall, $F1$-value, and AUC value as the evaluation parameters of the model. For the convenience of the following description, it is now assumed that the positive samples represent normal URLs and the negative samples represent malicious URLs. The precision represents the probability of actually being a positive sample in all predicted positive samples. The recall rate represents the probability of being predicted as a positive sample in the actual positive sample. The $F1$ value takes both precision and recall rate into account, allowing both to reach the maximum of the equilibrium. As shown in Figures 16(a) and 16(b), the precision-recall curve (P-R curve) and ROC curve based on the improved multilayer convolution recurrent neural network model are given, respectively.

As it is shown in Figure 16(a), the abscissa of the P-R curve represents the recall rate, and the ordinate represents the accuracy. With the increase of the recall rate of the model, more and more actual positive samples will be predicted as positive samples, while the model still has high accuracy. As it is shown in Figure 16(b), the abscissa of the ROC curve represents the proportion of false-positive samples to actual negative samples, and the ordinate represents the recall rate. When the proportion of false-positive samples predicted by the model decreases gradually, the model still has a high recall rate. It can be concluded that the improved multilayer convolutional recurrent neural network model proposed in this paper has the best classification effect and recognition ability. Finally, we, respectively, calculate the accuracy, recall rate, precision, $F1$-value, and AUC value of the six models under the test dataset. The results are shown in Figure 17.

It can be seen from Figure 17 that the improved multilayer convolutional recurrent neural network model is superior to the other five recognition models in accuracy,

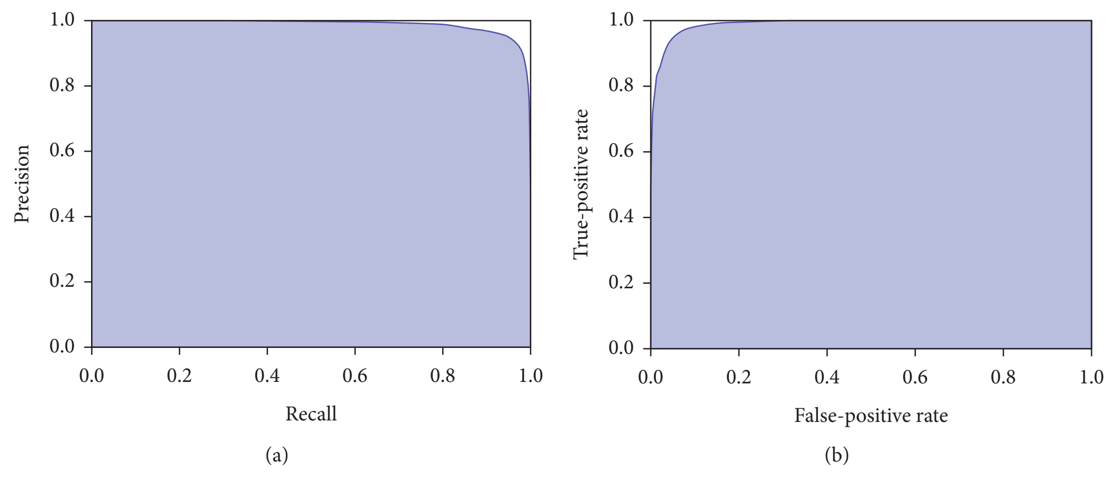


Figure 16: Model evaluation: (a) P-R curve and (b) ROC curve.

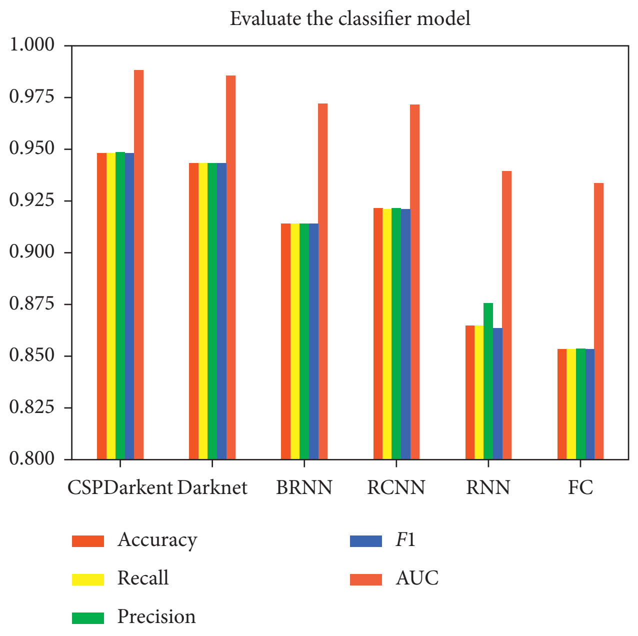


Figure 17: Model evaluation and comparison.

recall rate, precision, $F1$ value, and AUC value. Therefore, it can be concluded that the method proposed in this paper has higher recognition accuracy and better generalization ability than other existing malicious URL recognition models.

5. Conclusions

Malicious URL identification and detection are two of the important maintenance methods in maintaining network information security. The traditional malicious URL detection methods unduly rely on similarity matching rules, and the information of URL text is lost after numerical vectorization, which together makes it difficult to identify the context relationship of URL, and there are misjudgments and omissions. Therefore, this paper proposes an improved multilayer convolutional recurrent neural network model to detect malicious URLs. First, the model uses word embedding to participate in the training process of the entire neural network model, and the obtained word embedding space can vectorize the input URL text at the character level. Then, a single URL can be transformed into a two-dimensional tensor, and the feature extraction is carried out based on the multilayer convolutional neural network model improved by the YOLO algorithm. Finally, the extracted feature tensor is input into the bidirectional LSTM neural network for malicious URL recognition and detection, and the discriminant results are output. The experimental results show that when the embedding layer is used to participate in the training process of the entire model, the obtained embedding space has a relatively close relationship between the character vectors and at the same time has good representation ability. The CSPDarknet network improved based on the YOLO algorithm can effectively extract the features of the URL two-dimensional numerical tensor. At the same time, through the multilayer convolutional neural network, it can reduce the dimensionality of the URL numerical tensor and reduce the complexity of the model. The bidirectional LSTM recurrent neural network is used to process the convolution numerical tensor, which can effectively find the relationship between the contexts in the URL and further improve the detection accuracy of malicious URL. Through the evaluation of the model, it can be concluded that the improved multilayer convolution recurrent neural network model proposed in this paper can effectively improve the detection efficiency and recognition accuracy of malicious URLs by network security personnel and ensure the information security of network users. It also has a good prospect in the field of network security maintenance. However, since this paper adopts the truncated method to standardize the length of all URLs, it is inevitable to lose some information when facing a longer URL. Therefore, in the process of URL text vectorization, how to avoid missing information still has certain research value.

Data Availability

The data used to support the findings of this study are available at https://github.com/faizann24/Using-machine-learning-to-detect-malicious-URLs.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research has been partially supported by grants from the National Natural Science Foundation of China (Ref. 61903137). The work was also supported by grants from the National Key R&D Program of China for International S&T Cooperation Projects (2019YFE0118700) and the Natural Science Foundation of Hunan Province (Ref. 2020JJ5201).

References

[1] M. Sameen, K. Han, and S. O. Hwang, "PhishHaven-an efficient real-time AI phishing URLs detection system," *IEEE Access*, vol. 8, pp. 83425–83443, 2020.

[2] S. Anwar, F. Al-Obeidat, A. Tubaishat et al., "Countering malicious URLs in internet of things using a knowledge-based approach and a simulated expert," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4497–4504, 2020.

[3] X. Yan, Y. Xu, B. Cui, S. Zhang, T. Guo, and C. Li, "Learning URL embedding for malicious website detection," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6673–6681, 2020.

[4] T. Li, G. Kou, and Y. Peng, "Improving malicious URLs detection via feature engineering: linear and nonlinear space transformation methods," *Information Systems*, vol. 91, pp. 1–17, 2020.

[5] L. Vu, P. Nguyen, and D. Turaga, "Firstfilter: a cost-sensitive approach to malicious URL detection in large-scale enterprise networks," *IBM Journal of Research and Development*, vol. 60, no. 4, pp. 4:1–4:10, 2016.

[6] W. Yang, W. Zuo, and B. Cui, "Detecting malicious URLs via a keyword-based convolutional gated-recurrent-unit neural network," *IEEE Access*, vol. 7, pp. 29891–29900, 2019.

[7] J. Yuan, G. Chen, S. Tian, and X. Pei, "Malicious URL detection based on a parallel neural joint model," *IEEE Access*, vol. 9, pp. 9464–9472, 2021.

[8] P. Yang, G. Zhao, and P. Zeng, "Phishing website detection based on multidimensional features driven by deep learning," *IEEE Access*, vol. 7, pp. 15196–15209, 2019.

[9] H.-H. Wang, L. Yu, S.-W. Tian, Y.-F. Peng, and X.-J. Pei, "Bidirectional LSTM malicious webpages detection algorithm based on convolutional neural network and independent recurrent neural network," *Applied Intelligence*, vol. 49, no. 8, pp. 3016–3026, 2019.

[10] Z. Chen, M. Lu, Y. Zhou, and C. Chen, "Information synergy entropy based multi-feature information fusion for the operating condition identification in aluminium electrolysis," *Information Sciences*, vol. 548, pp. 275–294, 2021.

[11] R. Liao, R. Zhang, J. Guan, and S. Zhou, "A new unsupervised binning approach for metagenomic sequences based on *N*-grams and automatic feature weighting," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 11, no. 1, pp. 42–54, 2014.

[12] F. Jafarzadehpour, A. Sabbagh Molahosseini, A. A. Emrani Zarandi, and L. Sousa, "Efficient modular adder designs based on thermometer and one-hot coding," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 9, pp. 2142–2155, 2019.

[13] Z. Li, F. Yang, and Y. Luo, "Context embedding based on Bi-LSTM in semi-supervised biomedical word sense disambiguation," *IEEE Access*, vol. 7, pp. 72928–72935, 2019.

[14] L. Lu, Y. Yi, F. Huang, K. Wang, and Q. Wang, "Integrating local CNN and global CNN for script identification in natural scene images," *IEEE Access*, vol. 7, pp. 52669–52679, 2019.

[15] S. Wang, R. Yao, T. A. Tsiftsis, N. I. Miridakis, and N. Qi, "Signal detection in uplink time-varying OFDM systems using RNN with bidirectional LSTM," *IEEE Wireless Communications Letters*, vol. 9, no. 11, pp. 1947–1951, 2020.

[16] J. Ma, H. Liu, C. Peng, and T. Qiu, "Unauthorized broadcasting identification: a deep LSTM recurrent learning approach," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 9, pp. 5981–5983, 2020.

[17] T. Dai, L. Zhu, Y. Wang, and K. M. Carley, "Attentive stacked denoising autoencoder with bi-LSTM for personalized context-aware citation recommendation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 553–568, 2020.

[18] S. Albahli, N. Nida, A. Irtaza, M. H. Yousaf, and M. T. Mahmood, "Melanoma lesion detection and segmentation using YOLOv4-DarkNet and active contour," *IEEE Access*, vol. 8, pp. 198403–198414, 2020.

[19] M. M. Kalayeh and M. Shah, "Training faster by separating modes of variation in batch-normalized models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 6, pp. 1483–1500, 2020.

[20] J. Naranjo-Alcazar, S. Perez-Castanos, I. Martín-Morató, P. Zuccarello, F. J. Ferri, and M. Cobos, "A comparative analysis of residual block alternatives for end-to-end audio classification," *IEEE Access*, vol. 8, pp. 188875–188882, 2020.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 9953509, 11 pages
https://doi.org/10.1155/2021/9953509

Research Article

Creating Ensemble Classifiers with Information Entropy Diversity Measure

Jiangbo Zou,[1,2] Xiaokang Fu,[3,4] Lingling Guo,[5] Chunhua Ju,[3] and Jingjing Chen[6]

[1]*Chinese Academy of International Trade and Economic Cooperation, Beijing 10071, China*
[2]*School of Business Administration, Zhejiang Gongshang University, Hangzhou 310018, China*
[3]*School of E-Commerce & Management Science, Zhejiang Gongshang University, Hangzhou 310018, China*
[4]*Sunyard System Engineering Co., Ltd., Hangzhou 310053, China*
[5]*College of Chemical Engineering, Zhejiang University of Technology, Hangzhou 310014, China*
[6]*Zhijiang College, Zhejiang University of Technology, Shaoxing 312030, China*

Correspondence should be addressed to Jingjing Chen; joyjchan@gmail.com

Received 25 March 2021; Revised 20 April 2021; Accepted 9 May 2021; Published 24 May 2021

Academic Editor: Xiaokang Zhou

Ensemble classifiers improve the classification accuracy by incorporating the decisions made by its component classifiers. Basically, there are two steps to create an ensemble classifier: one is to generate base classifiers and the other is to align the base classifiers to achieve maximum accuracy integrally. One of the major problems in creating ensemble classifiers is the classification accuracy and diversity of the component classifiers. In this paper, we propose an ensemble classifier generating algorithm to improve the accuracy of an ensemble classification and to maximize the diversity of its component classifiers. In this algorithm, information entropy is introduced to measure the diversity of component classifiers, and a cyclic iterative optimization selection tactic is applied to select component classifiers from base classifiers, in which the number of component classifiers is dynamically adjusted to minimize system cost. It is demonstrated that our method has an obvious lower memory cost with higher classification accuracy compared with existing classifier methods.

1. Introduction

The ensemble method was firstly proposed by Hansen and Salamon to optimize neural networks [1]. It is well known that an ensemble learning model is usually more accurate than a single learning model [2–8]. According to Singh's work, classifier combination is now widely applied in the area of machine learning and pattern recognition, such as text classification, speech recognition, seismic wave analysis, communication network, and online transaction log analysis [9]. Instead of constructing a monolithic system, ensemble learning is used to construct a pool of learners and combine them in a smart way into an overall system. In the research area of dynamic data stream classification, ensemble learning has become one of the hot spots [10].

Recently, a great number of researches propose various kinds of classifiers especially in the field of data stream mining [11, 12]. To cope with concept drift, some published papers focus on dynamic weight mechanism. Previously, Wang and Pineau proposed online cost-sensitive boosting algorithms for online ensemble algorithms, which can achieve the similar accuracy of traditional boosting with simpler base models [13]. Tennant et al. presented a real-time data stream classifier to address the overlap of the velocity and volume aspects of big data analytics, which is adaptive to concept drift [14].

These ensemble classification approaches have good stability and can overcome the general concept drift phenomenon in data stream classification. However, there is no evidence that an ensemble classifier system with more single base classifiers is better than the ensemble classifier system with fewer single base classifiers. Sometimes, the classifier fusion method creates large-scale classifiers that require a great deal of memory and computing resources, leading to low efficiency. To solve this problem, Zhou offered valuable sights about the diversity metric, which can be leveraged to select a subset of learners to comprise the final ensemble [15]. It was proved that ensemble classifiers with greater diversity have stronger generalization ability. However, Bi demonstrated that the accuracy of classifiers was not strongly correlated with the diversity; in some contexts, the relationship was negative [16]. Luo put forward a self-adapted classifier ensemble method with particle classification information, considering both ensemble classifier accuracy and diversity [17]. In this model, particle classification information was used to mark the learning effect, and the product of weighted accuracy and diversity was the selection criteria in a based classifier filter named C-Lib. Thus, whether diversity correlated with ensemble performance is still unclear.

In this paper, we propose a method for generating ensemble classifiers by measuring the diversity between base classifiers, coming up with an incremental classification algorithm to maximize the diversity of component classifiers as well as minimizing the system cost of an ensemble classifier. We verify the approach in data stream mining along with other traditional algorithms, suggesting that the proposed model is efficient and promising.

2. Materials and Methods

2.1. Ensemble Classifier Diversity. The diversity of an ensemble classifier is the difference of base classifiers, and an ensemble classifier with high diversity means complementariness. According to the previous work, data misclassified by one classifier can be probably correctly classified by others, leading to higher overall performance and better stability of an ensemble classifier with several different base classifiers than that of a single classifier [18, 19].

Figure 1(a) presents the diversity between two linear classifiers in an ensemble classifier with different data distributions. Suppose Dataset A and Dataset B are datasets from two different classes, and the data distributions are denoted by two anomalous curves. Linear classifier p and linear classifier q are two selected base classifiers of an ensemble classifier, which are trained by test dataset. The correctly classified data by linear classifier p is denoted by the regions marked with horizontal bars: areas S1 and S2. The correctly classified data by linear classifier q is denoted by the regions marked with vertical bars: area H1. It is clear that there are still many blank regions left without correction and the diversity between the two linear classifiers is not high, resulting in ineffective ensemble classifiers.

Suppose we select another two base classifiers, linear classifier i and linear classifier j, which are combined to an ensemble classifier with the same data distribution in Figure 1(a). Figure 1(b) shows a better classification result. More data is correctly classified as shown in areas of S1′, S2′, H1′, and H2’. Comparing with the two ensemble classifiers, it is noted that the ensemble linear classifier with base classifier set (i, j) is significantly superior to the ensemble linear classifier with the base classifier set (p, q), which may attribute to the base classifier selection and optimization [20, 21].

In data stream mining, the distribution of the dataset changes rapidly over time. In Figure 1(c), the dataset is changed to dataset M and dataset N that are two typical data stream datasets, and the base classifiers of the ensemble classifier remain the same. It is suggested that the proportion of blank regions in Figure 1(b) increases and the accuracy of the ensemble classifier (i, j) for the dataset (A, B) is lower than that for the dataset (M, N). Such classification performance is far from the requirements for dynamical stream data mining. Therefore, the diversity of the same ensemble classifiers can be different when the dataset changed. The diversity measure method is particularly important in classifier combination optimization for better selection decision support, as well as the low computing resource consumption, especially in data stream classifiers.

Further, suppose another base linear classifier E is added to the ensemble classifier in Figure 1(b). If all the data in dataset A and dataset B can be correctly classified by the new ensemble classifier with E without blank area, such ensemble classifier is considered as the best ensemble classifiers and the diversity is positively correlated with ensemble classification accuracy. Instead, if the blank area is bigger than that in Figure 1(b), the performance of the new ensemble classifier with another base classifier F is lower and the diversity among base classifiers is negatively correlated with ensemble classification accuracy.

2.2. Diversity Measure Method in Ensemble Classifiers. Diversity among the members of a team of classifiers is deemed to be a key issue in the classifier ensemble problem. Unfortunately, diversity measurement is not straightforward because there is no generally accepted definition [14, 22–24]. According to Zhukov et al. [11], the diversity measure methods for ensemble classifiers can be divided into two categories: pairwise measure and nonpairwise measure. Pairwise diversity measures emphasis on local optimum calculates the average (dis) similarity metric between all possible pairs of individual classifiers in an ensemble, such as Q-statistic and correlation coefficient. Nonpairwise measure emphasizes on global optimum, which often calculates a statistic using the notion of entropy or using (dis) similarity metrics between individual classifiers and the averaged classifier [25–27]. Both methods combine accuracy and diversity together.

Relevant concepts are defined to describe the two types of diversity measures as follows: let $Z = \{z_1, \ldots, z_N\}$ be a training dataset with labels with M different classes in total, $z_j \in \mathfrak{R}^n$ coming from the classification problem in question. Let $D = \{D_1, D_2, \ldots, D_M\}$ be a set of base classifiers, D_i an N-dimensional binary vector, and vector $C = \{1, 2, \ldots, M\}$ the class label set. Assume z_j is a sample of training data from dataset Z, $z_j = \{A1, A2, \ldots, AS, Cj\}$, descried by s features value A and one class label value C_j belongs to C. The output of a base classifier Di for z_j is denoted $y_{j,} i = 1$, if D_i classified z_j to a class correctly, and 0; otherwise, $i = 1, \ldots, L$ by an N-dimensional binary vector $y_i = [y_{1,} i, \ldots, y_{N,} i]^T$. So, come

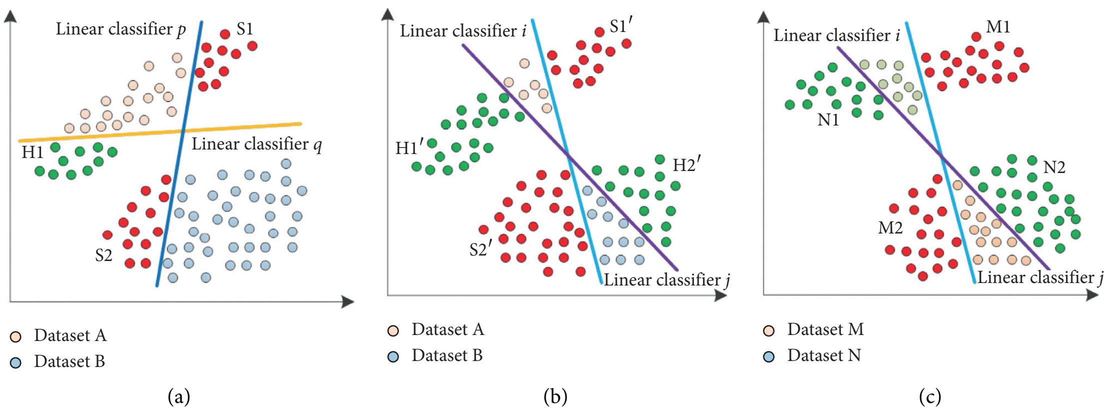


Figure 1: (a) Linear ensemble classifier; (b) linear ensemble classifier with higher diversity; and (c) linear ensemble classifier in data stream classification.

up the output matrix $Y = [y_{j,}\ i]^{LxM}$, including all of the classifying results from the training dataset Z and base classifiers set D. Let D_i and D_k be a pair of base classifiers from D; the relationship between them can be described as Table 1.

N^{ab} means the amount of training data that can be correctly classified by base classifier D_i, D_k or not. For example, N^{10} represents the amount of training data samples which are correctly classified by base classifier D_i, which are incorrectly classified by D_k. The table is from the conception of the confusion matrix. The size of training dataset Z is N, obviously, $N = N^{11} + N^{10} + N^{01} + N^{00}$, and two commonly used measures of diversity will be given as follows.

According to Yule's Q-statistic, the diversity between two base classifiers D_i and D_k can be calculated by the equation:

$$Q_{ik} = \frac{N^{11}N^{00} - N^{10}N^{01}}{N^{11}N^{00} + N^{10}N^{01}}, \quad (1)$$

where N^{ab} is the number of elements z_j of Z for which $y_{i,}\ j = a$ and $y_{j,}\ k = b$ (see Table 1). Q_{ik} ranges between −1 and 1; classifiers that classify more common objects correctly have a positive Q value. In contract, those that classify more objects to different classes will result in a negative Q value. If two base classifiers are statistically independent, the expectation of Q_{ik} is 0 [6, 28].

The correlation coefficient between two base classifiers can be calculated as follows:

$$\rho_{ik} = \frac{N^{11}N^{00} - N^{10}N^{01}}{\sqrt{\left(N^{11} + N^{10}\right)\left(N^{01} + N^{00}\right)\left(N^{11} + N^{01}\right)\left(N^{10} + N^{00}\right)|}}, \quad (2)$$

ρ_{ik} has the range as Q_{ik}, and they have the same changing trend. It can be proved that $|\rho_{ik}| < |Q_{ik}|$ [10]. For this comparison, diversity that measures by Q-statistic is more accurate and sensitive than that by the correlation coefficient.

In addition to these two pairwise measures of diversity, there are many other methods. The disagreement measure and the double-fault measure are two popular measures. In processing data stream by ensemble classifiers, pairwise diversity measure is an effective way to incrementally adjust the number of base classifiers. However, this paper applies nonpairwise diversity measures to classifier ensemble in the processing data stream, because nonpairwise measures can ensure the global optimal among classifiers when learning ensemble classifier. In this paper, information entropy is incorporated into the diversity measure. Entropy is defined as a measure of uncertainty in information theory; the greater the entropy value, the smaller the information uncertain degrees, and vice versa. Information entropy can be applied to the diversity measures of nonpairwise classifiers through the transformation of entropy.

Table 1: A 2×2 table of the relationship between a pair of classifiers.

	D_k correct (1)	D_k wrong (0)
D_i correct (1)	N^{11}	N^{10}
D_i wrong (2)	N^{01}	N^{00}
Total, $N = N^{00} + N^{01} + N^{10} + N^{11}$		

For a data sample Z_j, $Z_j \in Z$, the output of base classifier Di for the training data Z_j is denoted by y_{ji}. If Z_j is successfully classified by D_i, $y_{ji} = 1$, and otherwise 0, $i = 1, \ldots, L$. If the outputs of $|L/2|$ of the L base classifiers for Z_j are the same (0 or 1), the outputs of the left $L - L/2$ of the L base classifiers are the alternative value, coming up to the highest diversity among classifiers for Z_j. If all the y_{ji} values of the L base classifiers are the same, 0 s or all 1 s, there is no disagreement among base classifiers, coming up to the lowest diversity among classifiers for Z_j. For N training data, the measure of diversity based on information entropy is as the following equation:

$$E = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{(L - L/2)}\min\left\{R\left(Z_j\right), L - R\left(Z_j\right)\right\}. \quad (3)$$

In equation (3), $R(Z_j)$ denotes the number of classifiers from D with the same output value y_{ij}, and entropy E varies between 0 and 1, where 0 indicates no difference and 1 indicates the highest possible diversity among the base classifiers in D. In the context of data stream mining, E equals 0 means the lowest diversity among the base classifiers, and the number of base classifiers in the ensemble classifier can be reduced due to the reasonable classifier effectiveness. In contrast, the E value close to 1 means the diversity of the classifiers is high; several new base classifies can be added to the ensemble classifier for better classification effectiveness. Based on the above concepts, we design an incremental classification algorithm based on information entropy diversity measures to optimize the effectiveness of ensemble classifiers data stream processing.

2.3. An Incremental Classification Algorithm Based on Information Entropy Diversity Measure. A typical data stream processing flow chart is shown in Figure 2. A data stream is inputted in an incremental ensemble classifier continuously chronologically. The data stream is processed according to the time period and the time granularity, which is set based on different requirements. For example, the weblog data stream frequently changes so a fine time granularity is required. However, for the credit-rating data stream, a wide time granular can be accepted.

In the time period from $[t–f]$ to $[t]$, ensemble classifier $L_t–_f$ deals with coming data which arrive during the f times period, while at the time $[t]$, the model will be incrementally updated coming with a new ensemble classifier L_t to process data during $[t]$ to $[t+f]$. In order to make an ensemble model to prevent concept drift when processing data stream, an incremental process is necessary which can be achieved by iterating the process of updating the model in each time period.

Taking time $[t]$ for example, the training dataset of L_t is mainly composed of labeled data, which has already been classified by ensemble classifier $L_{t–f}$ in the period of $[t–f]$ to $[t]$. First, base classifiers are generated from the labeled training dataset by selected classification algorithms. Second, a certain number of base classifiers are selected to combine an incremental ensemble classifier L_t at time of $[t]$. The base classifiers in the new ensemble classifier are selected from ensemble new learning classifiers and old classifiers. The selection is based on two criteria, accuracy and diversity, which are measured by transformed information entropy. On one hand, we use accuracy as a criterion to remove base classifiers which have poor classification performance. On the other hand, the diversity criterion is used to adjust the number of base classifiers to achieve the global optimization of incremental ensemble classifier [29–31].

2.4. Incremental_SEM Algorithm. The most important process in generating an incremental ensemble classifier is selecting the most suitable classifiers with great accuracy and a proper number of classifiers. In this paper, the basic tactic for base classifier selection is integrating information entropy measure to the cyclic iterative selection algorithm, along with the accuracy performance data. The pseudocode of the base classifier selection algorithm for the proposed incremental classification model is given in Algorithm 1 Incremental_SEM.

Incremental_SEM uses cyclic iterative optimization selection method to maximize the information entropy difference and dynamically adjust the number of ensemble classifiers. The key part of the algorithm lies in the setting of the interval threshold of classification diversity, which should be set according to different applications. Since the initialization and preprocessing part is the same as the traditional method of the processing data stream, it is skipped in the paper. Starting from computing the diversity of ensemble classifier $L_{t–f}$, we compare its value to the interval threshold and take different actions according to the comparison (line 3). If the value is higher than the upper limit of the interval threshold, keep generating a new base classifier and add it into the ensemble classifier. Recompute the diversity of a new ensemble classifier until the diversity is located in the interval threshold (lines 4–13). If the value is lower than the lower limit of the interval threshold, compute the accuracy of each base classifier and kick out the base classifier with the lowest accuracy (lines 14–19). Otherwise, if the value is located in the interval threshold, it is no need to update the ensemble classifier for the next time stage (lines 21–23).

3. Results

This section lists the experiments conducted to evaluate the performance of the proposed algorithm on data stream classification. Trace based simulation approach has been used to evaluate and compare the performance of the proposed algorithm with other baseline algorithms.

3.1. Experimental Data. The proposed algorithm was evaluated on steam data generated by a massive online analysis (MOA) system. MOA is a software environment for implementing algorithms and running experiments for online learning from evolving data streams. We select the following stream generators to generate data.

(i) Hyperplane generator generates a problem of predicting the class of a rotating hyperplane. HP1 and HP2 are the data stream generated by hyperplane generator with 5% noise data in the experiment.

(ii) Random tree generator generates a random radial basis function stream. It constructs a decision tree by choosing attributes at random to split and assigning a random class label to each leaf. RT1 and RT2 are data stream generated by random tree generator with both label attribute and number attribute.

(iii) SEA generator generates SEA concept functions. This dataset contains abrupt concept drift. SEA1 is the data stream generated by SEA generator with 5% noise data and concept drift.

(iv) STAGGER generator generates STAGGER Concept functions which were introduced by Schlimmer. SG1 is generated by STAGGER generator.

A detailed description of the experimental data stream is shown in Algorithm 1. Due to the infinite nature of data

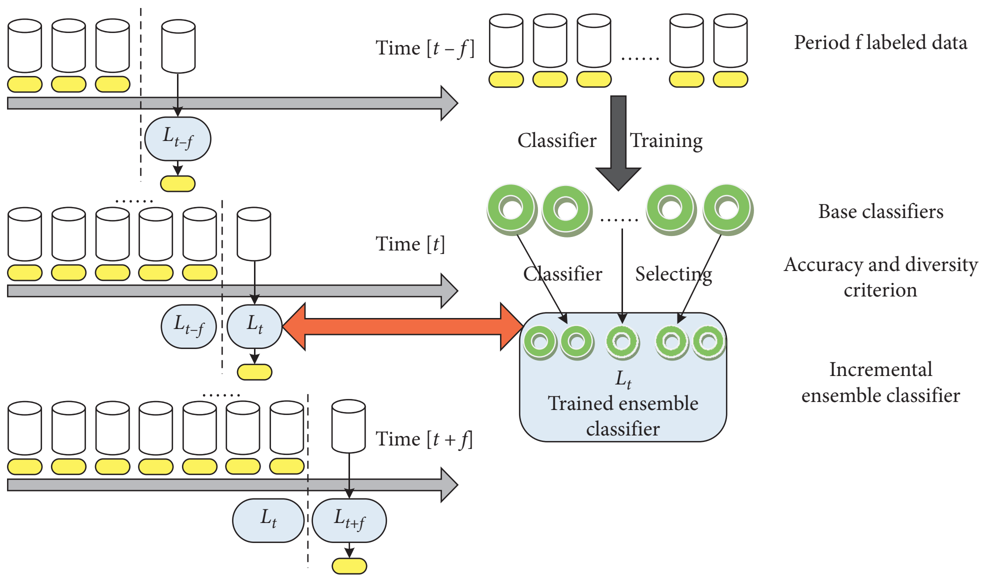


Figure 2: Incremental model of processing data stream.

stream in a real environment, it is not easy to do simulations in experiments. Massive data is used to simulate infinite data stream, the experiment data size of each dataset is shown in column 2 of Table 2.

Figure 3 shows the scatter diagrams of each dataset that helps understand data more intuitively. Since the volume of each dataset is large, partial data is selected to be shown in the diagrams. Usually, a dimension reduction operation is needed for the preprocessing dataset. As shown in Figure 3, it can be found that each attribute is nonlinear relativity.

3.2. Experiment Setup. An open-source mining software, WEKA, has been used to realize the ensemble classifier algorithms. The baseline algorithms in Weka are Naïve Bayes, Sequential Minimal Optimization (SMO), J48 that is the implementation of C4.5 for building a decision tree, IBk that is the implementation of the *K*-nearest neighbor algorithm (KNN), Kstar that is an instance-based classifier, NNge, PART that builds a '"partial" C4.5 decision tree in each iteration and makes the "best" leaf into a rule and AOD [32, 33]. The algorithms are shown in Table 3.

A computer with 1.73 GHz CUP and 2 G memory is used as the experiment computer, installed with the operating system Windows XP. In order to study the effectiveness of the proposed approach, experiments were setup to compare Incremental_SEM with Bagging and AdaBoost on different datasets. In all ensemble methods, decision trees were used as the base classifier. Based on WEKA 3.6, the decision tree construction method was J48 from the Weka library, which are selected to generate base classifiers with the default parameter sets [10]. The performance of each ensemble classifier was evaluated using a stratified 10-fold cross-validation procedure, in which the original dataset was partitioned randomly into 10 equal size subsamples and each fold contains roughly the same proportions of class labels. The experiment settings were as follows: the parameters of Bagging and AdaBoost were kept at their default values in Weka. The ensemble size can be regarded as a hyperparameter of the ensemble method. It can be tuned through cross-validation or using a separate validation set. It can also be thought of as an indicator of the operating complexity of the ensemble. For Incremental_SEM, different information entropy intervals were set for the six generated datasets, interval [0.21, 0.43] for HP1, HP2 and SEA1, [0.63, 0.85] for SG1, and [0.46, 0.69] for RT1 and RT2.

3.3. Results and Analysis. As shown in Figure 2, f is set as a time interval in the incremental model of the processing data stream, and the classifier is adjusted every time period. For each algorithm, the accuracy of the current ensemble classifier was calculated in every time period. We verified algorithms from two aspects: classification accuracy and system memory cost. Suppose at time t, ensemble classifier has m base classifiers; each base classifiers classification accuracy is $a_i (i = 1, 2, \ldots m)$. Take At as ensemble classification accuracy: At = (a1 + a2 + + am)/m. The ensemble classification results of the dataset in Algorithm 1 are shown in Figure 4.

In Figure 4(a), it is clear that the accuracy of the Incremental_SEM algorithm is slightly higher than the Bagging algorithm and both of them are obviously higher than a single algorithm when comparing the experiment results of Incremental_SEM algorithm with Bagging and Single classifier in datasets HP1 and RT1 at the time interval value of 10 seconds. However, the execution time of Incremental_SEM algorithm and bagging is longer than the single classifier, mainly because diversity computing in Incremental_SEM is time-consuming. Moreover, in order to test the memory cost while adding entropy diversity in ensemble classifiers, Incremental_SEM with traditional incremental algorithms, bagging, and without diversity measure are investigated. The experiment results are shown

```
  Input:
 Training dataset with labels by ensemble classifier $L_{t-f}$;
 The interval threshold of classification diversity: $[a, b]$;
 Iterate number (each iterate creates a new base classifier: $k$)
 Ensemble classifier at period of $[t\_f, t]$: $L_{t-f}$;
Output: incremental ensemble classifier $L_t$ at time $t$.
 (1)     Begin
 (2)     Loop
 (3)        Compute diversity value $\lambda_0$ of ensemble classifier $L_{t-f}$;
 (4)        If $\lambda_0 \in [b, 1]$
 (5)           For $i = 1$ to $k$
 (6)              Sampling training data from labeled dataset at period of $[t-f, t]$ by $L_{t-f}$;
 (7)              Generate a new base classifier $L_i$;
 (8)              Add $L_i$ to $L_{t-f}$;
 (9)              Compute the diversity value $\lambda_1$;
(10)           If $\lambda_1 \in [a, b]$
(11)                 $L_t = L_{t-f}$ ;
(12)                 Return $L_t$
(13)        End for
(14)     else if $\lambda_0 \in [0, a]$
(15)        Compute the accuracy of each base classifier at $L_{t-f}$;
(16)        Sort base classifiers in decreasing order of accuracy as baselist;
(17)        Delete some member base classifiers with the lowest accuracy at $L_{t-f}$;
(18)        Update the $L_{t-f}$;
(19)        $L_t = L_{t-f}$;
(20)        return $L_t$;
(21)     else
(22)           $L_t = L_{t-f}$;
(23)           return $L_t$
(24)     End if
(25)     Break;
(26)  End loop
```

Algorithm 1: Incremental_SEM algorithm.

in Table 4 with the average classification accuracy (ACA) and average system memory cost (ASM).

In Figure 4(b), it is noted that Incremental_SEM classification accuracy is almost the same as AdaBoost algorithm and both of them are higher than a single classifier when comparing Incremental_SEM with AdaBoost classification algorithm in datasets HP2 and RT2 at the time interval value of 20 seconds. Due to the higher dimension of the two datasets, the algorithm executing time average is longer than that in Figure 4(a), illustrating that learning a new base classifier is time-consuming. The results support the conclusion that adding diversity can increase classification time without improvement of entire effectiveness through comparing Incremental_SEM with a single algorithm.

In Figure 4(c), it is clearly noted that sharp accuracy drops (such as at times 30, 55, 60, 75) since the concept drift phenomenon existed in both two datasets when comparing Incremental_SEM with the AdaBoost algorithm at the time interval value of 5 seconds. Comparing with a single algorithm, Incremental_SEM and AdaBoost are more stable, suggesting that ensemble classifier has an advantage when concept drift exists in the dataset. It can be concluded that adding diversity into the ensemble classifier can improve algorithm performance, which is consistent with the view by Nan and Zhou [34–37].

From Tables 4–6, it can be found that the accuracy of Incremental_SEM classification is not significantly higher than AdaBoost and Bagging algorithm and all of them are nearly the same in our experiment. However, the average system memory cost of Incremental_SEM is much lower than AdaBoost and Bagging. It can be demonstrated that, for system memory cost, Incremental_SEM classification is better than traditional ensemble classification algorithms.

In order to testify the advantages of adopting entropy as a diversity measure when processing data stream, an experiment with the dataset in Table 3 was conducted to compare Q-statistic with correlation coefficient diversity measure. Table 7 shows that Incremental_SEM average accuracy is higher with Q-statistic than that with correlation coefficient ρ.

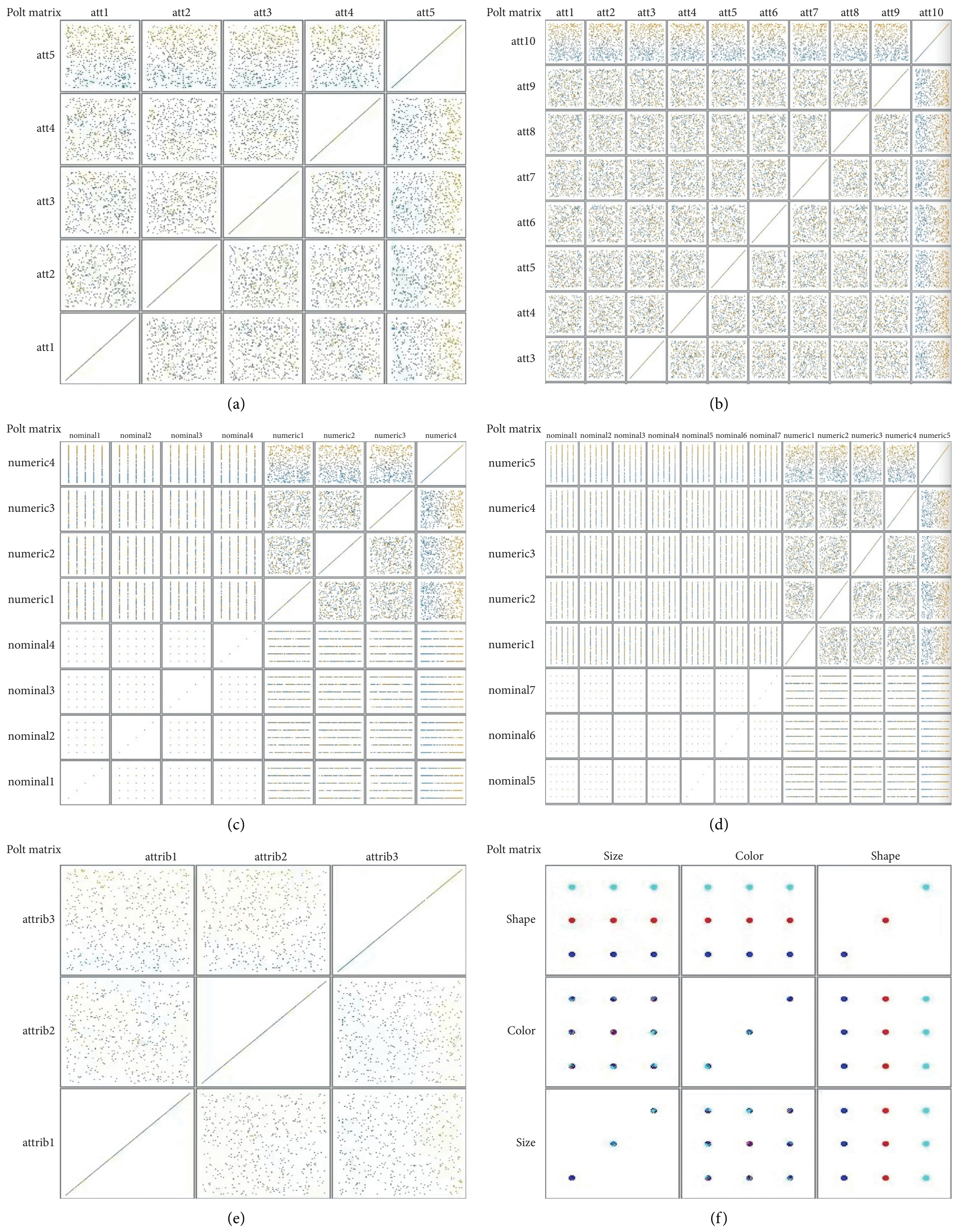
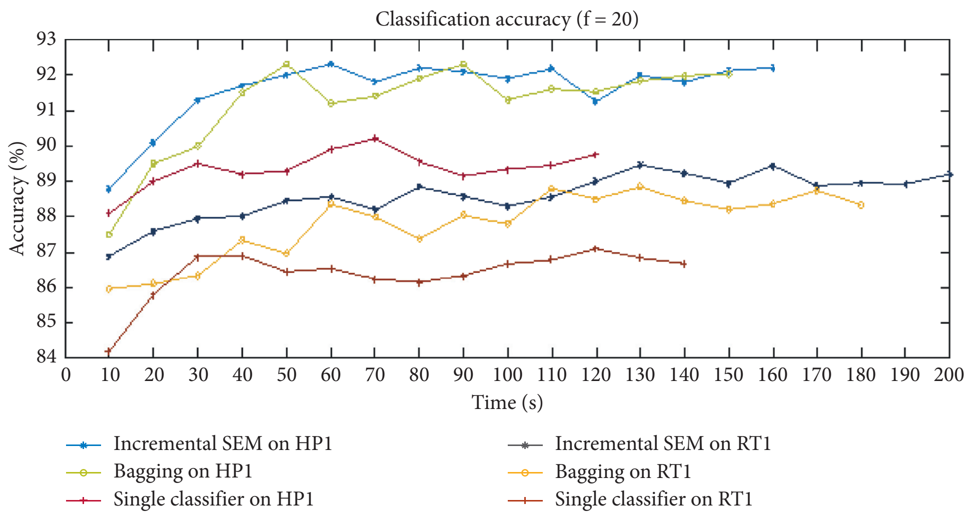


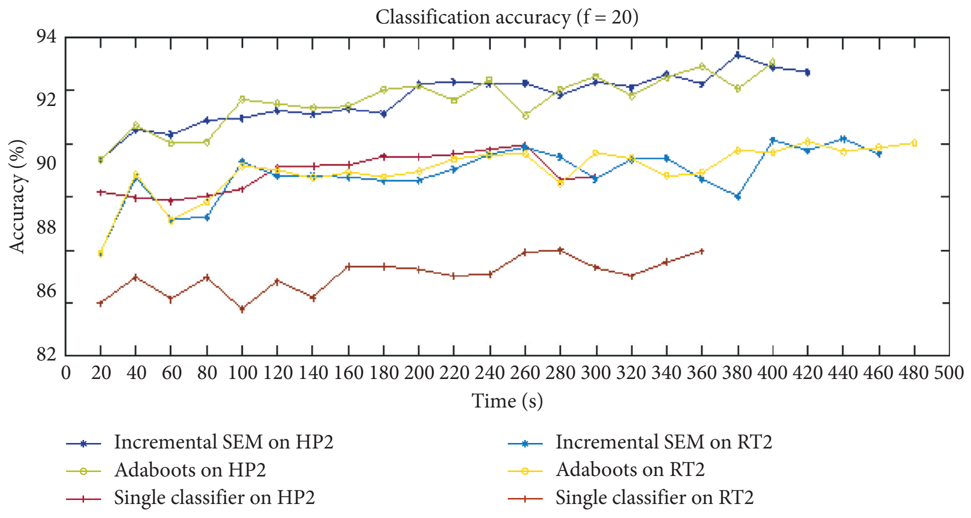
Figure 3: The scatter diagrams of the dataset. (a) HP1. (b) HP2. (c) RT1. (d) RT2. (e) SEA1. (f) SG1.

Table 2: Dataset description.

Dataset name	Data size	Classification number	Attribute number (label attribute/number attribute)
HP1	200,000	10	5 (0/5)
HP2	400,000	5	10 (0/10)
RT1	200,000	8	8 (4/4)
RT2	400,000	4	12 (7/5)
SEA1	500,000	2	3 (0/3)
SG1	500,000	2	3 (3/0)

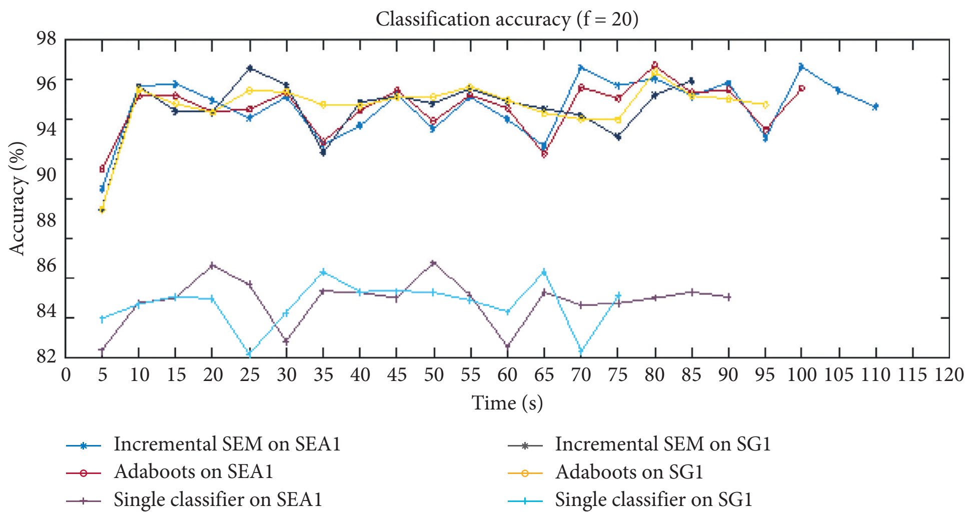


(a)



(b)

Figure 4: Continued.



(c)

FIGURE 4: The comparison experiments on (a) dataset HP1 and RT1, (b) dataset HP2 and RT2, and (c) dataset SEA1 and SG1.

TABLE 3: A list of classification algorithms available by Weka.

Number	Classifier name	Simple description of classifiers
1	Naïve Bayes	The Naïve Bayes classifier using kernel density estimation over multiple values for continuous attributes, instead of assuming a simple normal distribution
2	SMO	Sequential minimal optimization algorithm for training a support vector classifier using polynomial kernels
3	J48	Decision tree, the implementation of C4.5
4	IBk	An instance-based learning algorithm, the implementation of k-nearest neighbor algorithm (kNN)
5	KStar	The *K* instance-based learner using all nearest neighbors and an entropy-based distance
6	NNge	Nearest neighbor-like algorithm using nonnested generalized exemplars
7	PART	Generating a PART decision list for classification
8	AOD	Perform classification by averaging over all of a small space of alternative Naive Bayes-like models that have weaker independence

TABLE 4: The comparison of HP1 and RT1.

		HP1	RT1 (%)
Incremental_SEM	ACA	91.608%	88.599
	ASM	217MB	344
Bagging	ACA	91.19%	87.805
	ASM	250MB	485
Single	ACA	89.372	86.398
	ASM	157MB	237

TABLE 5: The comparison of HP2 and RT2.

		HP2 (%)	RT2 (%)
Incremental_SEM	ACA	91.732	88.861
	ASM	271	292
AdaBoost	ACA	91.677	88.943
	ASM	315	358
Single	ACA	88.859	84.976
	ASM	186	198

Table 6: The comparison of SEA1 and SG1.

		SEA1 (%)	SG1 (%)
Incremental_SEM	ACA	94.697	94.798
	ASM	106	97
AdaBoost	ACA	94.606	94.671
	ASM	138	115
Single	ACA	85.042	84.946
	ASM	78	82

Table 7: Comparison experiment between Q-statistic and correlation coefficient ρ.

	HP1 (%)	HP2 (%)	RT1 (%)	RT2 (%)	SEA1 (%)	SG1 (%)
Q-statistic	1.939	1.516	1.943	2.445	0.915	1.263
ρ	1.629	1.478	2.089	2.038	0.917	0.986

4. Conclusions

Ensemble classifier, as a common algorithm at processing data stream, is famous for its high classification accuracy and stability. We proposed an ensemble algorithm incorporating entropy as the diversity measure. It is proved that our Incremental_SEM algorithm has a higher classification accuracy rate than a single classifier and lower system memory cost than the Bagging and AdaBoost algorithm. It is also suggested that the Q-statistic diversity measure outperforms the correlation coefficient diversity measure. Future research will focus on how to verify the relativeness between accuracy and diversity in theory.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request as the size of the experimental data is too large to upload via this submission interface.

Conflicts of Interest

The authors declare no conflicts of interest.

Authors' Contributions

J.Z. and J.C. conceptualized the study; X.F. was responsible for methodology; J.Z., L.G., and J.C. validated the study; X.F. contributed to formal analysis; L.G. investigated the study; X.F. provided resources; J.Z. was responsible for data curation; J.Z. prepared the original draft; J.C. reviewed and edited the manuscript; C.J. was responsible for project administration. All the authors have read and agreed to the published version of the manuscript.

Acknowledgments

This research was funded by Zhejiang Gongshang University Youth Program Project (grant no. 3090JYN9920001G-332). The funders had no role in the design of the study; the collection, analyses, or interpretation of data; the writing of the manuscript, or the decision to publish the results.

References

[1] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[2] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Machine Learning*, vol. 51, no. 2, pp. 181–207, 2003.

[3] L. I. Kuncheva, "Diversity in classifier ensembles," in *Combining Pattern Classifiers: Methods and Algorithms*John Wiley and Sons, Hoboken. NJ, USA, 2004.

[4] F. Schwenker, F. Roli, and J. Kittler, "Multiple classifier systems," in *Proceedings of the 12th International Workshop, MCS 2015*, Günzburg, Germany, July 2015.

[5] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, "A survey on ensemble learning for data stream classification," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 23–32, 2017.

[6] D. V. R. Oliveira, G. D. C. Cavalcanti, and R. Sabourin, "Online pruning of base classifiers for dynamic ensemble selection," *Pattern Recognition*, vol. 72, pp. 44–58, 2017.

[7] M. Muzammal, R. Talat, A. H. Sodhro, and S. Pirbhulal, "A multi-sensor data fusion enabled ensemble approach for medical data from body sensor networks," *Information Fusion*, vol. 53, pp. 155–164, 2020.

[8] X. Zhou, W. Liang, S. Shimizu, J. Ma, and Q. Jin, "Siamese neural network based few-shot learning for anomaly detection in industrial cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5790–5798, 2021.

[9] P. K. Singh, R. Sarkar, and M. Nasipuri, "Correlation-based classifier combination in the field of pattern recognition," *Computational Intelligence*, vol. 34, no. 3, pp. 839–874, 2018.

[10] M. Tennant, F. Stahl, O. Rana, and J. B. Gomes, "Scalable real-time classification of data streams with concept drift," *Future Generation Computer Systems*, vol. 75, pp. 187–199, 2017.

[11] A. V. Zhukov, D. N. Sidorov, and A. M. Foley, "Random forest based approach for concept drift handling," in *Proceedings of the 2017 International Conference on Analysis of Images, Social Networks and Texts*, Moscow, Russia, July 2017.

[12] C. M. Salgado, S. M. Vieira, L. F. Mendonça, S. Finkelstein, and J. M. C. Sousa, "Ensemble fuzzy models in personalized medicine: application to vasopressors administration," *Engineering Applications of Artificial Intelligence*, vol. 49, pp. 141–148, 2016.

[13] B. Wang and J. Pineau, "Online bagging and boosting for imbalanced data streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 12, pp. 3353–3366, 2016.

[14] Z. Zhou, J. Wu, and W. Tang, “Ensembling neural networks: many could be better than all,” *Artificial Intelligence*, vol. 137, no. 12, pp. 239–263, 2002.

[15] X. Zhou, W. Liang, Z. Luo, and Y. Pan, “Periodic-aware intelligent prediction model for information diffusion in social networks,” *IEEE Transactions on Network Science and Engineering*, 2021.

[16] J. Luo and D. Chen, “Application of adaptive ensemble algorithm based on correctness and diversity,” *Journal of Zhejiang University (Engineering Science)*, vol. 45, no. 3, pp. 558–562, 2011.

[17] K. Yan, A. Chong, and Y. Mo, “Generative adversarial network for fault detection diagnosis of chillers,” *Building and Environment*, vol. 172, Article ID 106698, 2020.

[18] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera, “A survey on data preprocessing for data stream mining: current status and future directions,” *Neurocomputing*, vol. 239, pp. 39–57, 2017.

[19] M. Mojirsheibani and C. Shaw, “Classification with incomplete functional covariates,” *Statistics & Probability Letters*, vol. 139, pp. 40–46, 2018.

[20] K. Yan, J. Su, J. Huang, and Y. Mo, “Chiller fault diagnosis based on VAE-enabled generative adversarial networks,” *IEEE Transactions on Automation Science and Engineering*, 2020.

[21] X. Zhou, X. Xu, W. Liang et al., “Intelligent small object detection based on digital twinning for smart manufacturing in industrial CPS,” *IEEE Transactions on Industrial Informatics*, p. 1, 2021.

[22] J. Obregon, A. Kim, and J.-Y. Jung, “RuleCOSI: combination and simplification of production rules from boosted decision trees for imbalanced classification,” *Expert Systems with Applications*, vol. 126, pp. 64–82, 2019.

[23] L. Guo, Q. Liu, K. Shi, Y. Gao, J. Luo, and J. Chen, “A blockchain-driven electronic contract management system for commodity procurement in electronic power industry,” *IEEE Access*, vol. 9, pp. 9473–9480, 2021.

[24] J. Lu, D. Chen, G. Wang, D. Kiritsis, and M. Törngren, “Model-based systems engineering tool-chain for automated parameter value selection,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–15, 2021.

[25] P. Porwik, R. Doroz, and K. Wrobel, “An ensemble learning approach to lip-based biometric verification, with a dynamic selection of classifiers,” *Expert Systems with Applications*, vol. 115, pp. 673–683, 2019.

[26] X. Zhou, Y. Hu, W. Liang, J. Ma, and Q. Jin, “Variational LSTM enhanced anomaly detection for industrial big data,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3469–3477, 2021.

[27] X. Zhou, Y. Li, and W. Liang, “CNN-RNN based intelligent recommendation for online medical pre-diagnosis support,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, p. 1, 2020.

[28] J. Xia, M. Dalla Mura, J. Chanussot, P. Du, and X. He, “Random subspace ensembles for hyperspectral image classification with extended morphological attribute profiles,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 9, pp. 4768–4786, 2015.

[29] J. Y. Choi, D. H. Kim, K. N. Plataniotis, and Y. M. Ro, “Classifier ensemble generation and selection with multiple feature representations for classification applications in computer-aided detection and diagnosis on mammography,” *Expert Systems with Applications*, vol. 46, pp. 106–121, 2016.

[30] K. Yan, L. Liu, Y. Xiang, and Q. Jin, “Guest editorial: AI and machine learning solution cyber intelligence technologies: new methodologies and applications,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6626–6631, 2020.

[31] Z. Zhang, Y. Zeng, and K. Yan, “A hybrid deep learning technology for PM 2.5 air quality forecasting,” *Environmental Science and Pollution Research*, pp. 1–14, 2021.

[32] X. Zhou, W. Liang, K. I.-K. Wang, R. Huang, and Q. Jin, “Academic influence aware and multidimensional network analysis for research collaboration navigation based on scholarly big data,” *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 1, pp. 246–257, 2021.

[33] S. B. Kotsiantis, “Supervised machine learning: a review of classification techniques,” in *Proceedings of the 2007 conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, Amsterdam, Netherlands, June 2007.

[34] L. Nan and Z. Zhou, “Selective ensemble of classifier chains,” in *Proceedings of the 2013 11th International Workshop on Multiple Classifier Systems, MCS 2013*, Nanjing, China, May 2013.

[35] G. D. C. Cavalcanti, L. S. Oliveira, T. J. M. Moura, and G. V. Carvalho, “Combining diversity measures for ensemble pruning,” *Pattern Recognition Letters*, vol. 74, pp. 38–45, 2016.

[36] Y. Bi, “The impact of diversity on the accuracy of evidential classifier ensembles,” *International Journal of Approximate Reasoning*, vol. 53, no. 4, pp. 584–607, 2012.

[37] N. Jin, Y. Zeng, K. Yan, and Z. Ji, “Multivariate air quality forecasting with nested LSTM neural network,” *IEEE Transactions on Industrial Informatics*, 2021.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 9999398, 14 pages
https://doi.org/10.1155/2021/9999398

Research Article

Towards Efficient Video Detection Object Super-Resolution with Deep Fusion Network for Public Safety

Sheng Ren, Jianqi Li, Tianyi Tu, Yibo Peng, and Jian Jiang

School of Computer and Electrical Engineering, Hunan University of Arts and Science, Changde 415000, China

Correspondence should be addressed to Jianqi Li; hnwlpyb@huas.edu.cn

Received 22 March 2021; Revised 14 April 2021; Accepted 14 May 2021; Published 24 May 2021

Academic Editor: David Megías

Video surveillance plays an increasingly important role in public security and is a technical foundation for constructing safe and smart cities. The traditional video surveillance systems can only provide real-time monitoring or manually analyze cases by reviewing the surveillance video. So, it is difficult to use the data sampled from the surveillance video effectively. In this paper, we proposed an efficient video detection object super-resolution with a deep fusion network for public security. Firstly, we designed a super-resolution framework for video detection objects. By fusing object detection algorithms, video keyframe selection algorithms, and super-resolution reconstruction algorithms, we proposed a deep learning-based intelligent video detection object super-resolution (SR) method. Secondly, we designed a regression-based object detection algorithm and a key video frame selection algorithm. The object detection algorithm is used to assist police and security personnel to track suspicious objects in real time. The keyframe selection algorithm can select key information from a large amount of redundant information, which helps to improve the efficiency of video content analysis and reduce labor costs. Finally, we designed an asymmetric depth recursive back-projection network for super-resolution reconstruction. By combining the advantages of the pixel-based super-resolution algorithm and the feature space-based super-resolution algorithm, we improved the resolution and the visual perception clarity of the key objects. Extensive experimental evaluations show the efficiency and effectiveness of our method.

1. Introduction

Video surveillance systems are widely distributed in urban streets and roads, commercial places, residential areas, bank outlets, stations, terminals, airports, and other public places, playing an increasingly important role in public security. Through the video surveillance system, suspicious signs and objects can be found in time and monitored closely, to avoid the occurrence of criminal harm effectively. The police can obtain information about criminals by surveillance videos and inquire about the locations of suspected vehicles and personnel. In the interrogation stage of case investigation, the surveillance video can be used as objective litigation evidence. Video surveillance has become the fourth major field of investigation technology after criminal science and technology, action technology, and network investigation technology. It plays an irreplaceable role in the construction of a safe and smart city. In 2015, the Ministry of Public Security, the Ministry of Science and Technology, and other nine ministries and commissions put forward "several opinions on strengthening the networking application of public security video monitoring construction". And, they pointed out that constructing the network application of public security video surveillance helps to maintain national security and social stability and to prevent and crack down on violent terrorist crimes under the new situation. It is of great significance to improve the urban and rural management and innovate the social governance system.

The traditional video surveillance system can only provide real-time monitoring or manually analyze cases by reviewing the surveillance video. It leads to a low usage rate of surveillance video data. With artificial intelligence and machine learning technologies, video surveillance systems can intelligently analyze video content, detect abnormal behaviors, and discover potentially harmful behaviors [1]. Besides, these technologies can assist police and security personnel in investigating cases, thereby providing more accurate and safer surveillance. Alibaba Cloud's intelligent video surveillance platform can identify fireworks and inspectors wearing helmets, detect intruders in the area, and escort safe production. Baidu's video surveillance development platform EasyMonitor has a wealth of AI business skills, including electronic fences, smoke detection, safety cap detection, and departure detection [2]. Large Internet companies have released many intelligent video surveillance products. However, on the one hand, limited by the cost of the software product purchase, operation, and maintenance, the surveillance video field lacks simple and effective auxiliary tools. On the other hand, limited by hardware cost, hardware technology, and shooting environment, the current surveillance video suffers from low-resolution and unclear visual perception. In summary, there are two problems in the field of public security based on surveillance video: (1) in most cases, surveillance video viewers need to manually identify the object and manually select video keyframes for analysis, making it easy to lose the object in addition to being inefficient. (2) It is difficult to use the low-resolution video frames since it is easy to lose high-frequency information when zooming in to view the object, resulting in blurring and hardness of recognition.

To solve the above problems, we proposed a super-resolution method for video detection objects. We use object detection algorithms to assist surveillance video viewers to track objects in real time and use super-resolution algorithms to reconstruct high-resolution video frames with clear visual perception. The traditional object detection algorithm OpenCV cascade classifier uses a sliding window to select the region, then uses HOG + SVM and other methods for feature extraction, and finally uses the classifier to classify the detection area [3, 4]. Object detection algorithms based on deep learning can be divided into two types: object detection and recognition algorithms based on region recommendations and object detection; recognition algorithms based on regression. R-CNN, object detection and recognition algorithm based on region recommendations, first generates object candidate frames based on region recommendations, then filters the generated object candidate frames, and finally refines the size and position of the candidate frames [5]. The regression-based object detection and recognition algorithm YOLO regards object detection as a regression problem. The training phase aims to train a set of weights and directly call the trained weights for object positioning during testing [6]. Traditional super-resolution methods are mainly based on interpolation (such as zero-order interpolation, bilinear interpolation, and bicubic interpolation) and examples. The instance-based sparse representation method establishes the mapping relationship between low-resolution and super-resolution images by learning the sparse associations between image blocks to achieve super-resolution reconstruction of images [7]. Super-resolution algorithms based on deep learning can be divided into pixel-space-based methods and feature-space-based methods. A super-resolution method based on pixel space SRCNN first uses a $9 * 9$ convolutional layer to extract the initial features of the image, then uses a $1 * 1$ convolutional layer to learn the nonlinear mapping from low resolution (LR) to high resolution (HR), and finally uses a $5 * 5$ convolutional layers reconstruct super-resolution images [8]. SRGAN, a super-resolution method based on feature space, learns the nonlinear mapping from LR to HR through a generator, and then the discriminator constrains the generated super-resolution image in terms of semantics and style [9]. These object detection algorithms and super-resolution algorithms have achieved better and better results in their respective fields, but they lack integration, unification, and optimization for specific application scenarios and are not suitable for direct use in the field of public security based on video surveillance [10].

In this paper, we proposed a super-resolution method based on a deep fusion network for surveillance video object detection. Firstly, we designed a comprehensive surveillance video analysis framework that integrates object detection algorithms, keyframe selection algorithms, and super-resolution algorithms. It solves the problem of large workload, easy object loss, and low resolution in public security video data analysis. Secondly, we used regression-based object detection and recognition algorithms to identify video objects in real time, which is convenient for surveillance video viewers to track objects. Besides, we employed the keyframe selection algorithm to select the frames with significant changes in the scene of the surveillance video to reduce the workload of video analysis. Finally, a super-resolution algorithm combining pixel space and feature space is used to reconstruct the object in the keyframe. It is beneficial to improve the resolution of key objects and the visual perception quality of objects detected by surveillance video and surveillance video viewer's investigation and handling cases. The main contributions of this paper are summarized as follows: (1) We designed a novel comprehensive analysis framework for surveillance video. It improves the efficiency and accuracy of video analysis by combining object detection, keyframe selection, and super-resolution algorithms. (2) We proposed a keyframe selection algorithm and use regression-based object detection and recognition algorithm to recognize video objects in real time. (3) We proposed a super-resolution approach that deeply integrates the advantages of pixel space and feature space to improve the resolution of surveillance video detection objects.

The rest of this paper is organized as follows: Section 2 explains the related works, Section 3 describes the work process of our approach, Section 4 explains our approach in detail, and Section 5 provides experimental results. Section 6 concludes our work.

2. Related Work

2.1. Object Detection Algorithms. Object detection is the basic task of computer vision and can be widely used in object tracking, crowd counting, face recognition, and other fields. It is an important algorithm in public safety. Object detection algorithms based on candidate regions, also known as two-stage object detection algorithms, mainly include region-based convolution neural networks (R-CNN), Fast R-CNN, Faster R-CNN, and other models

[11–13]. Regression-based object detection algorithms, also known as one-stage object detection algorithms, mainly include YOLO series algorithms and SSDs [14, 15]. The two-stage object detection algorithm consists of two steps. We first generate object candidate frames based on regional suggestions, then filter the candidate frames, and classify the objects. Faster R-CNN first uses the Region Proposal Network (RPN) to generate a candidate frame and then uses softmax to determine whether the candidate frame is in the foreground or the background. And, it employs bounding box regression to correct the candidate frame to obtain a more accurate candidate frame, called proposals. Then RPN uses the region of interest (ROI) layer pool proposals of different sizes into the same size and uses a fully connected layer for object classification and position adjustment regression [5]. The two-stage object detection algorithm has high recognition accuracy, but the generation and selection of candidate frames will consume a lot of computing power and time, and it is difficult to reach the detection speed of 24FPS, which is not suitable for real-time object detection. YOLO v5 directly returns the position and category of the bounding box in the output layer. It first uses Mosaic data enhancement to solve the problem of many small objects in the data set and uneven distribution on the input side. Then, it introduces the focus and cross stage partial (CSP) structure into backbone to improve the feature extraction capabilities. Besides, in neck, it uses feature pyramid network (FPN) and pyramid attention network (PAN) structures to fuse semantic features and positioning features of feature maps at different scales and aggregates parameters for different detection layers. Finally, on the prediction side, it uses generalized intersection over union (GIoU) loss to handle the noncoincidence of bounding boxes. YOLO v5 has a faster running speed and can meet the requirements of real-time object detection. The object detection algorithm will generate a lot of redundant information in the surveillance video frames. Selecting keyframes from abundant video frames for efficient object tracking is an effective method to improve detection efficiency and reduce the calculation and workload.

2.2. Super-Resolution Algorithms. The super-resolution algorithm can reconstruct the corresponding high-resolution image from a single or a series of low-resolution images. It is a low-level computer vision algorithm and the basis of a high-level computer vision algorithm. The super-resolution algorithms based on deep convolutional neural networks can reconstruct high-quality super-resolution images with rich high-frequency information and clear texture features, which have become the mainstream research methods. The super-resolution convolutional neural networks (SRCNN) use convolutional neural networks for the first time in the field of super-resolution and only use three-layer convolutional neural networks to surpass most traditional super-resolution methods [8]. Compared with the example-based super-resolution method, SRCNN does not require complicated preprocessing and can optimize the super-resolution reconstruction results several times through backpropagation, which not only improves the quality of reconstructed results but also improves the efficiency. The main disadvantage of SRCNN is that it is difficult to build deep convolutional neural networks to improve feature extraction and characterization capabilities. The residual network ResNet connects the input side information to the output side via a shortcut connection [16]. The convolutional layer only needs to learn the residuals of the input and output, which is the basis for building a deep neural network. The key to super-resolution reconstruction is to optimize high-frequency information. The residuals learned by the residual network contain rich high-frequency information, and since most of the residuals are close to 0, the learning efficiency is very high. Therefore, the residual network plays a very important role in improving the efficiency and quality of super-resolution reconstruction and becomes the backbone network of the super-resolution methods. Super-resolution using very deep convolutional networks (VDSR) introduces the residual network into the super-resolution field for the first time and builds a super-resolution network with a depth of 20 layers, which expands the receptive field and improves the convergence speed [17]. The speed and quality of VDSR are significantly better than SRCNN, and it uses a single network to reconstruct super-resolution images of different multiples. Megvii Research Institute proposed a method that can achieve super-resolution reconstruction at any multiple through a single model [18]. Zhang et al. of Harbin Institute of Technology proposed a plug-and-play super-resolution method based on regular depth to process low-resolution images with arbitrary blur kernels [19]. SenseTime proposed a super-resolution method of real scenes with original images [20]. Li et al. of Sichuan University proposed an image super-resolution feedback network to improve the low-level representation with high-level information [21]. Li et al. of Wuhan University proposed a fast spatiotemporal residual network for video super-resolution [22]. Gu et al. of the Chinese University of Hong Kong proposed a fuzzy kernel estimation method in the blind oversegmentation problem [23]. Dai et al. of Tsinghua University proposed a second-order attention network for image super-resolution [24], and Ma et al. proposed a method of constraining the super-resolution image structure using a gradient map [25]. Xu et al. of China Taiwan MediaTek Inc. proposed a dynamic convolutional network to realize super-resolution restoration of multiple combinations of blur kernels and noisy images [26]. The super-resolution method based on pixel space produces more photo-realistic images, while the high-frequency information generated by the super-resolution method based on feature space is rich. By combining the advantages of the two methods in a deep fusion network, we can reconstruct high-quality key objects.

To solve the problem that video surveillance system in the field of public security lacks basic and intelligent management and analysis algorithms, we proposed a super-resolution method of surveillance video objects based on a deep fusion convolutional neural network. It assists police and other surveillance video analysts to track, identify, and analyze objects and investigate cases. By optimizing the regression-based one-stage object detection algorithm, we can identify the objects in the surveillance video in real time, which assists the viewers to track the video objects and analyze the video content. We designed a keyframe selection algorithm for surveillance video. By analyzing the object category, number, and confidence degree in the video frame, a small number of video frames with significant changes in the object are selected from a large number of videos to assist the surveillance video viewer to quickly locate the objects and reduce the workload. We used the super-resolution algorithm based on the deep fusion network to reconstruct the determined object. It helps to improve the resolution of the key object, assist the surveillance video viewer to carefully check the details of the key object, and improve the quality of the surveillance video content analysis. The super-resolution method of surveillance video objects based on a deep fusion convolutional neural network constructed in this paper can be effectively used in the field of public security to assist police and security personnel to track and analyze surveillance video objects. And, it also provides an effective auxiliary tool for preventing and cracking down on violent terrorist crimes.

3. Preliminary

3.1. Object Detection and Super-Resolution Process. The regression-based one-stage object detection method takes the video frame as the input of the network and returns the position and category of the bounding box (BBox) at the output end. For each grid of the video frame, it predicts n BBox and c category information. Among them, each BBox contains four location information items (w, y, w, h) and one confidence information item. The BBox position information (x, y) is used to calibrate the center point of the BBox, and (w, h) is used to calibrate the width and height of the BBox relative to the video frame. Confidence predicts the confidence of the objects contained in the BBox and the accuracy of the BBox compared to the real object box. The confidence is defined by (1), where C means confidence, O means object (if there is an object, $\Pr(O) = 1$; otherwise $\Pr(O) = 0$), and $\mathrm{IOU}_{\mathrm{pre}}^{\mathrm{tru}}$ predicts the intersection over union between the BBox and the real box. Define A as the predicted frame and B as the real frame; S is the set of all frames; and $A, B \subseteq S \in \mathbb{R}^n$. IoU can be described by (2). According to the predicted category information of each grid of the video frame and the confidence of the BBox prediction, we can obtain the class-specific confidence score of each BBox, which is defined in (3). According to the class-specific confidence score, the BBox with a low score is filtered out by setting a threshold, and the remaining BBox is processed by non-maximum suppression (NMS) to obtain the final detection result.

$$C = \Pr(O) * \mathrm{IOU}_{\mathrm{pre}}^{\mathrm{tru}}, \quad (1)$$

$$\mathrm{IOU} = \frac{|A \cap B|}{|A \cup B|}, \quad (2)$$

$$CC = \Pr(\mathrm{Class}_i) * \Pr(O) * \mathrm{IOU}_{\mathrm{pre}}^{\mathrm{tru}}, \quad (3)$$

$$I_{\mathrm{LR}} = (I_{\mathrm{HR}} * k)\downarrow_s + n, \quad (4)$$

$$I_{\mathrm{LR}} = \mathrm{bicubic}(I_{\mathrm{HR}})\downarrow_s, \quad (5)$$

$$I_{\mathrm{SR}} = \mathrm{upsample}(\mathrm{Res}(\mathrm{Feature}(I_{\mathrm{LR}}))). \quad (6)$$

The super-resolution reconstruction method reconstructs a corresponding high-resolution image or video based on one or a series of low-resolution images (video frames). Since it is difficult to obtain a series of low-resolution images of the same reconstruction object, current research mainly focuses mainly on single-image super-resolution. Video super-resolution reconstruction needs to comprehensively utilize information in both space and time dimensions and improve the reconstruction effect of the center frame by using the information of adjacent low-resolution frames. The degradation process of an image or video frame can be described by (4), where I_{LR} represents a low-resolution image, I_{HR} refers to a high-resolution image, $\downarrow_s$ represents a downsampling scale, k is a blur kernel, and n represents noise. In real-world scenes, there are many types of downsampling scales, blur kernels, and noises. To facilitate supervised deep learning, the SR method generally uses bicubic interpolation to downsample high-resolution images and obtains paired images (LR, HR) for learning the mapping relationship from low resolution to high resolution. The process of super-resolution reconstruction is shown in (6). It can be typically divided into three steps. Firstly, we use the convolutional layer (Feature) to extract the low-resolution initial features. Then, we employ the residual network (Res) to learn the nonlinear mapping from low resolution to high resolution. Finally, we use the upsampling layer (Upsample) to enlarge the low resolution to a specified scale and reconstruct the super-resolution image.

3.2. Work Process Overview. Video surveillance systems widely distributed in communities, roads, streets, and commercial places (supermarkets) are a powerful guarantee of public safety. The combination of traditional video surveillance systems and artificial intelligence technologies can effectively improve the accuracy of video content analysis and abnormal behavior warning, greatly improve the work efficiency of surveillance video viewers, and reduce labor costs. The super-resolution framework of the video detection object based on deep learning is shown in Figure 1. By fusing the object detection algorithm based on deep learning and the super-resolution algorithm, we can track the object of the surveillance video in real time, pick out the keyframes that change significantly, and perform super-resolution reconstruction of the key object. This provides the police and judges with clear and high-resolution objects, which can assist in the investigation and review of related cases. The method proposed in this paper can solve the problems of traditional video surveillance, provide effective assistance to surveillance video viewers, and effectively serve the public security field.

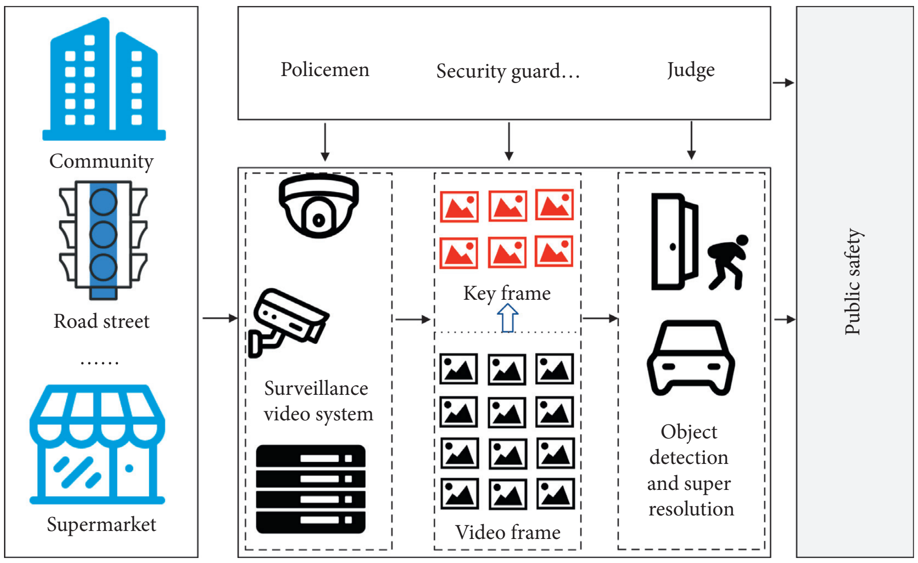


Figure 1: The work process of the video detection object super-resolution.

The super-resolution method of video detection object based on deep learning is mainly divided into three steps. Firstly, we use a regression-based object detection algorithm to perform real-time object detection on surveillance videos obtained from communities, roads, streets, supermarkets, and other places. Marking the scope, category, and confidence of the object in real time in the video frame assists police, security, and other video viewers to track the object. Then, we use a keyframe selection algorithm to select frames with significant changes from a large number of videos and select the key information from a large amount of redundant information to help viewers analyze video content quickly and efficiently. We finally use the super-resolution algorithm to reconstruct, improve the resolution of, and identify the key object.

4. Video Detection Object Super-Resolution

4.1. Video Detection Object Super-Resolution Network. To improve the safety of public places such as communities, supermarkets, roads, and streets and assist surveillance video viewers to quickly and effectively locate objects, we design a video object detection super-resolution algorithm that combines object detection and super-resolution reconstruction. As shown in Figure 2, we use a regression-based object detection algorithm for real-time object detection and a deep back-projection network for super-resolution reconstruction. It should be noted that, for the convenience of drawing, we did not show the details of each functional module in detail in Figure 2, such as the feature fusion module of different scales of object detection and the feature prediction module of video frames, and the asymmetric recursive structure of super-resolution.

The object detection network consists of three parts. Firstly, we use the backbone built by modules such as focus, CBL (conv, BN, Leaky ReLU), and CSP to extract video frame features. The focus module is used to slice the input image and transform the dimension of the feature map. For example, we can slice the $608 * 608 * 3$ image into a $304 * 304 * 12$ feature map according to RGB. Then, it passes through a conv layer containing 32 filters and transforms it into a $304 * 304 * 32$ feature map. The CBL module is composed of the conv layer, the BN layer, and the Leaky ReLU activation function, and CBL is the basic module unit of the object detection network. The CSP module is composed of CBL, residual unit, conv layer, BN layer, and Leaky ReLU activation function. It is the feature splitting module unit of the object detection network. Secondly, we use the multiscale fusion module constructed by CBL, CSP-1, CSP-2, and other modules to extract fusion features of different scales and strengthen the ability of network feature fusion (the structure of the CSP-1 and CSP-2 modules is shown in Figure 3, and the relevant descriptions are provided in Section 4.2). Finally, we use GIoU loss to solve the problem that the detection frame and the real frame do not intersect to optimize the detection frame and speed up the convergence of the model.

Before we perform super-resolution reconstruction, we will select the detection object to avoid wasting computing power to super-resolve a large number of worthless objects. The selection process of the object is described in detail in Section 4.2. Our super-resolution reconstruction algorithm uses a deep asymmetric recursive back-projection network to reconstruct key objects. The super-resolution network consists of three parts. Firstly, we employ two conv layers to extract the initial features of the image. Then, we use the upper and lower iterative sampling layers to learn the mapping relationship between the low- and high-resolution image several times to obtain upsampling features of different levels. We set up a recursive structure to deepen the network level and learn advanced semantic features without increasing parameters. By sharing the downsampling layer and setting an asymmetric structure, we reduce the parameters of the network model. We fuse homology residuals

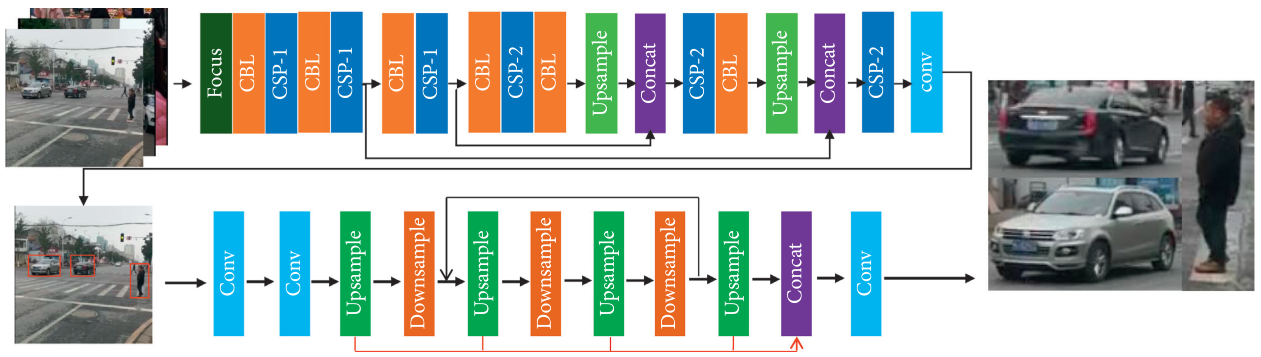


Figure 2: The video detection object super-resolution network.

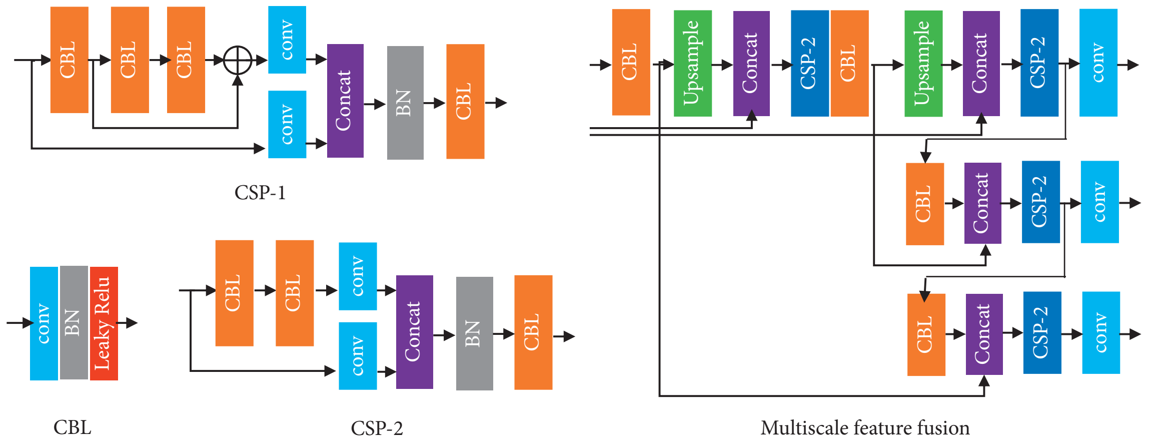


Figure 3: The core module of object detection network and multiscale feature fusion network.

and cascaded residuals in the upsampling unit to make full use of residual information while correcting projection errors. We concatenate cascade non-depth upsampling features to improve the high-frequency detail information of upsampling features [27, 28]. Finally, we use the conv layer to reconstruct the super-resolution video frame detection object.

4.2. Object Detection and Keyframe Selection Method. The object detection algorithm can assist police and security personnel in analyzing surveillance video objects and investigating security cases. It helps to track and monitor suspicious signs and objects in real time to avoid criminal harm. The object detection algorithm includes two core modules, CBL and CSP, in which CSP is composed of residual structure CSP-1 and convolution structure CSP-2, as shown in Figure 3. CBL is the basic module and consists of conv layer, BN layer, and Leaky ReLU activation function. The CSP-1 module includes two branches. The first branch includes a CBL feature extraction unit, two CBL residual mapping units, and a conv layer. The second branch directly uses a conv layer to extract the input low-level features, and then we fuse the output of the two branches through the concatenation layer. The CSP-2 module and the CSP-1 module share the same structure, and the difference is that the first branch of CSP-2 does not use the residual structure.

The multiscale feature fusion module uses the top-down FPN structure to amplify the deep feature information and fuse it with the backbone features of different depths using the upsampling unit. We use the bottom-up PAN structure to downsample the upsampled features and merge them with neck features of different depths. The FPN structure integrates strong semantic features from top to bottom, and PAN integrates strong positioning features from bottom to top. The multiscale feature fusion module structure is shown in Figure 3.

Our object detection algorithm first uses backbone to extract the features of the input video frame; then uses the multiscale feature fusion module to strengthen the ability of network feature fusion; and finally uses GIoU loss to optimize the object frame, filter the object frame through DIOU_NMS, and output object frame information, classification information, and confidence information. The objective box optimization loss function is $\mathrm{GIoU} = \mathrm{IoU} - |A_c - U|/|A_c|$, where A_c is the smallest bounding box between the predicted box A and the real box B, and U is the union of the predicted box and the real box $U = A \cup B$. GIoU can distinguish different positional relationships between predicted frames and real frames with the same IoU and the same size and can optimize the situation where the prediction frame and the real frame do not intersect. DIOU_NMS also considers the overlap area and the center distance between the two boxes when filtering the object box.

DIOU_NMS will not delete two boxes with a far center point since they may be in two different objects. DIOU_NMS instead of NMS can optimize the detection accuracy of overlapping objects. It can improve the recognition accuracy of occluded objects without increasing the computational cost. Real-time object detection can assist surveillance video viewers to track objects. However, it still takes a lot of labor costs to face the massive amount of video information. Therefore, selecting keyframes with significant scene changes can effectively improve the efficiency of key object recognition. The keyframe selection algorithm we designed is shown in Algorithm 1.

According to the output of the surveillance video object detection algorithm, we can select frames from the video whose category, number, and detection frame have significant changes compared with the previous frame as keyframes. Static pictures often appear in surveillance videos (the object category and amount in the video have not changed). Detecting object super-scores for all static video frames will result in a large amount of redundant information, which not only wastes computing power but also is not conducive to video content analysis. As shown in Algorithm 1, our keyframe selection algorithm can use information such as the type, quantity, and confidence of the detection output to compare and analyze the difference between the current frame and the previous frame while detecting the object. If the surveillance video currently captures a static image, we believe that this frame has less valuable information and will not store keyframes. If any of the following three situations occurs, the keyframe selection algorithm can select a small number of frames with significant changes from the surveillance video: (1) The object category of the current frame changes relative to the previous frame; for example, the object of the previous frame has cars and trees, but the object of the current frame contains cars, trees, and people. (2) Compared with the previous frame, the amount of a certain category in the current frame has changed; for example, the previous frame has cars (5 cars), trees (3), and people (1), and the current frame has cars (5 cars), trees (3), and people (2). (3) Compared with the previous frame, the detection box of the current frame has changed. For example, the detection box of a car in the previous frame is ($b_x = 0.2$, $b_y = 0.7$, $b_w = 0.1$, $b_h = 0.15$), and the detection box of the same car in current frame is ($b_x = 0.5$, $b_y = 0.5$, $b_w = 0.25$, $b_h = 0.4$). Video viewers can understand the surveillance video content by viewing a few frames to improve work efficiency. The time complexity of the whole algorithm is $O(T_1 + N)$, where T_1 is the time of video object detection and N is the number of video frames.

4.3. Super-Resolution Method of Video Detection Object.

The goal of the super-resolution method based on pixel space is to make the super-resolved image as close as possible to the real image at every pixel point in pixel space. These methods use L1 or L2 as the loss function and do not use adversarial training to generate video frames, and the reconstructed results are close to real-world video frames in pixel space. However, it is easy to lose high-frequency information, which leads to extremely smooth and fuzzy reconstructed video frames and poor visual perception quality. The feature space-based super-resolution method aims to make the feature space of the super-resolved image close to the real-world image. These approaches combine the perception loss, confrontation loss, and L1 and L2 as the loss function. They use adversarial training to generate video frames of high visual perception quality. However, these methods easily lead to deformation and distortion of super-resolution image structure. Therefore, we propose a super-resolution method based on the fusion of pixel space and feature space. The super-resolution method based on pixel space ensures the authenticity of video frames while the fusion of the loss function based on feature space improves the visual perception quality of video frames.

The super-resolution method based on the deep back-projection structure learns the mapping relationship between LR and SR several times in the reconstruction process through the up- and downsampling units placed in sequence. The structure of the up- and downsampling unit is shown in Figure 4. The upsampling unit includes 2 deconv layers and 1 conv layer. The first deconv layer enlarges the input LR feature map to a specified scale, and the conv layer transforms the enlarged feature map back to the original size and calculates the cascade projection error with the input LR of this unit and the homologous projection error with the original input LR. The second deconv layer amplifies the error information to a specified scale and adds it to the first deconv layer. It corrects the output of the upsampling unit by cascading and homologous projection errors. The downsampling unit includes 2 conv layers and 1 deconv layer. The structure and execution process of the downsampling units are similar to the upsampling unit. We did not use the homologous error in the downsampling unit and only used the cascaded error to correct the output of the downsampling unit.

To improve the effect of super-resolution reconstruction, we need to stack several upsampling and downsampling units to obtain high-level semantic features. However, stacking several upsampling units and downsampling units will result in a sharp increase of the model parameters, and the training and use of the model will become more difficult. As shown in Figure 4, we propose a recursive back-projection structure to increase the depth of the back-projection network without increasing the parameters. We input 2 sets of up- and downsampling units into the recursive loop structure and set the recursive hyperparameter to 8. We use the parameters of 2 upsampling and downsampling units to achieve the performance of 16 upsampling and downsampling units. Besides, we introduce an asymmetric structure based on recursive back-projection to further reduce the model parameters and improve the robustness of the model in practical applications. As shown in Figure 4, we concatenate the SR feature maps of upsampling units at different depths to reconstruct the output SR video frame to improve the effect of SR reconstruction. We use the downsampling units to transform the output of the upsampling units back to the original size. Therefore, we propose that all upsampling units share the same

```
    Input: $V$# input video
    Output: $F_{\text{KEY}}$# output keyframe
(1)   define$\mathcal{B}$as backbone of the object detection network
(2)   define$\mathcal{N}$ as neck of the object detection network
(3)   define$P$as prediction of the object detection network
(4)   define$\mathcal{U}$as output of the object detection network
(5)   $\mathcal{U} = \text{DIOU\_NMS}(P(\mathcal{N}(\mathcal{B}(V))))$
(6)   $\mathcal{U} \longrightarrow (C, N, B, P)$ # Choose category, quantity, BBox, confidence
(7)   for $i = 1,\ldots n$ do #$n$ is the number of video frames
(8)     if $(C_i != C_{i-1} or N_i != N_{i-1})$
(9)       if $(P_i > 0.5)$
(10)        Storage $(F_i)$
(11)  end for
(12)    $F_{\text{KEY}} \leftarrow (F_1, F_2, \ldots, F_k)$ #k is the number of key video frames
(13)  Return$F_{\text{KEY}}$;
```

Algorithm 1: Keyframe selection algorithm.

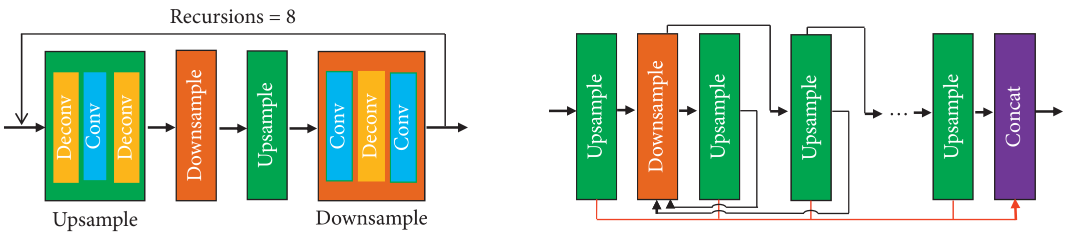


Figure 4: The sampling units, recursive structures, and asymmetric back-projection structure.

downsampling unit to construct a deep recursive back-projection network with an asymmetric structure.

$$L_T = \alpha L_1 + \beta L_{cx}, \tag{7}$$

$$L_1 = \frac{1}{N}\left\| \sum_{i=1}^{N} F_{\text{DBPN}}\left(I_i^{\text{LR}}\right) - I_i^{\text{HR}} \right\|_1, \tag{8}$$

$$L_{cx}\left(\phi\left(I^{\text{SR}}\right), \phi\left(I^{\text{HR}}\right), l\right) = -\log\left(CX\left(\phi^l\left(I^{\text{SR}}\right), \phi^l\left(I^{\text{HR}}\right)\right)\right). \tag{9}$$

The image reconstructed by the super-resolution method based on pixel space lacks high-frequency detail information, and the texture features are not clear enough. We tried to directly generate super-resolution images using methods based on generative adversarial networks. Then, we found that these approaches can improve the clarity of visual perception. However, the generated SR image suffered from structural deformation and distortion, it is prone to mode collapse when processing real-world video frame super-resolution reconstruction, and the robustness of the model is not enough. Therefore, we propose to use the L1 loss function based on the pixel space and the contextual loss based on the feature space on the asymmetric depth recursive back-projection network. The contextual loss function is shown in (9), where $\phi(I^{\text{SR}})$ represents the SR image feature map, $\phi(I^{\text{HR}})$ represents the HR image feature map, and ϕ uses a pretrained 19-layer VGG network. We select the feature map of the 5th layer before max-pooling after the 4th layer of convolution. We have $CX(x, y) = (1/N)\sum_j \max_i CX_{ij}$, where CX_{ij} is the similarity of local features x_i and y_j. We decouple the video frame into a collection of local features and optimize the reconstruction of I^{SR} features by measuring the distance between the local feature distributions of I^{SR} and I^{HR}. Based on the asymmetric recursive back-projection SR network, we add contextual loss to optimize the reconstruction of high-frequency information in the feature space. Combining the advantages of the authenticity of the pixel space SR method and the high visual perception quality of the feature space SR method, we can reconstruct a high-quality video frame object with realistic visual perception.

5. Experimental Results

5.1. Implementation Details. We use PyTorch to build a super-resolution model for video detection objects. In the model training phase, we use a high-performance server to train and verify the model. In the model testing phase, we use a personal computer for testing to ensure the usability and robustness of the model in actual application scenarios. The high-performance server runs on Linux operating system, and the GPU is NVIDIA TITAN Xp. The operating system of the personal computer for testing is Windows 10 with the i5 CPU core. We have taken some videos that do not involve personal privacy in communities, roads, streets, supermarkets, and schools. We then extracted the video frames for incremental training of the object detection module. We selected some high-resolution video frames from the shooting videos and added them to the DIV2K data set, and then we expanded the training set to 1,000 and the validation set and test set to 200.

$$\text{PRE} = \frac{\text{TP}}{(\text{TP} + \text{FP})}, \tag{10}$$

$$\text{REC} = \frac{\text{TP}}{(\text{TP} + \text{FN})}. \tag{11}$$

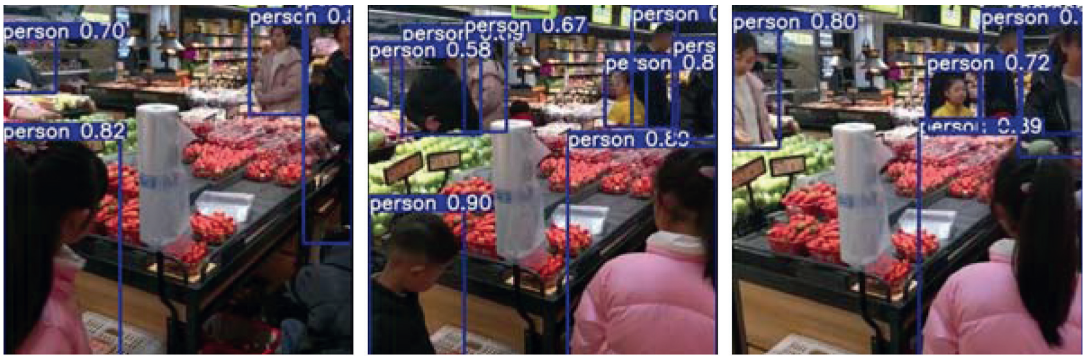
The evaluation metrics of the super-resolution method for video detection objects include the accuracy and speed of object detection, the peak signal-to-noise ratio, and the structural similarity of SR images. We use the mean Average Precision (mAP) to evaluate the object detection accuracy, which mainly includes the precision rate and the recall rate, as shown in (10) and (11). The numerator of precision is the number of detection frames with IoU greater than 0.5, and the denominator is the sum of the detection frames with IoU greater than 0.5 and less than 0.5. The numerator of recall is the number of detection frames with IoU greater than 0.5, and the denominator is the sum of detection frames with IoU greater than 0.5 and the true frames that are not detected. We use FPS (frames per second) to evaluate the object detection speed, which is the number of video frames that the model can process per second. We require the model to reach 30FPS on an ordinary PC. The PSNR is based on the sensitivity of pixel error, and we use it to measure the mean square error of the SR image and the real image. The unit of PSNR is dB. The larger the value, the smaller the distortion. The SSIM (structural similarity) measures the similarity of images from three aspects: brightness, contrast, and structure. The value range of SSIM is [0, 1]. The larger the value, the smaller the image distortion.

5.2. Object Detection Experiment Results. We first obtained some surveillance videos online, but we found that the scenes involved in these surveillance videos are relatively single. To verify the effectiveness of our method in the field of public safety, we have taken some surveillance videos that do not involve personal privacy in communities, roads, streets, supermarkets, schools, and other places. The video frame rate is 30FPS, the video resolution is 720p (16 : 9), and the 10-second video size is about 1.5 MB. We use regression-based object detection algorithms to detect objects in surveillance videos in real time. The object detection algorithm assists police and security personnel to track suspicious objects in real time and prevent crimes. On the other hand, the output information of object detection lays the foundation for the keyframe selection and key object super-resolution reconstruction. We use surveillance videos of three scenes: supermarkets, communities, and roads, to verify the effect of the object detection algorithm, as shown in Figure 5.

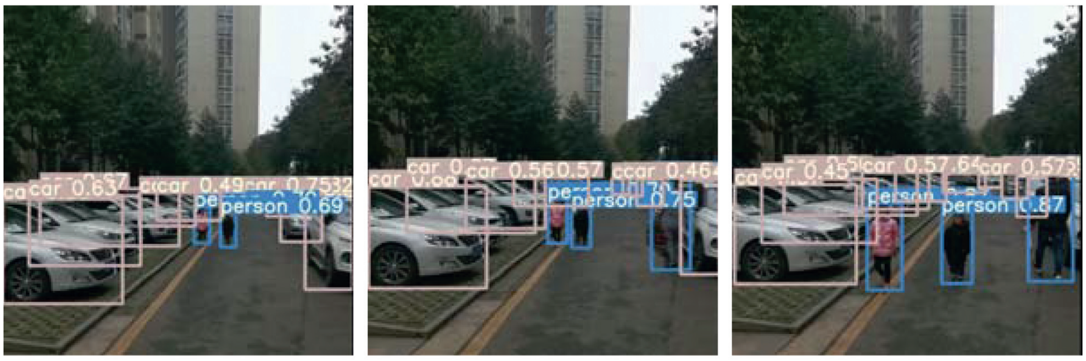
In the supermarket scene, the main detection object of surveillance video is people. Our object detection algorithm can accurately detect people in real time. We can monitor the valuables in the supermarket in real time by expanding the training data set. When the type, quantity, and confidence of the valuables change, we can promptly remind the supermarket management personnel. The real-time detection of people and valuables can assist supermarket managers to track suspicious persons in time and prevent the theft of valuables. In the community scenario, the main monitoring objects of surveillance video are people and cars, which helps in ensuring the safety of people and property. Since there are many small objects in community video surveillance and the occlusion between vehicles in the parking area is more serious, the object detection algorithm needs to recognize small objects and occluded objects well. We can see that our object detection algorithm can well identify small objects and obscured objects from Figure 5. It helps the community managers to track suspicious persons and vehicles in real time and maintain community safety. In road scenes, the main monitoring objects of surveillance video are people, vehicles, etc. We use object detection algorithms to assist traffic police to track suspicious people and vehicles in real time and maintain public safety.

5.3. Keyframe Selection Experiment Results. When investigating criminal cases, the police often need to find a very small number of frames with key information from a large number of videos. Selecting the keyframes manually requires a lot of costs and material resources and is inefficient. Based on the object detection algorithm, we designed a keyframe selection algorithm to pick out keyframes with significant changes in category, number, and confidence from a large amount of redundant video frame information. We selected 12 keyframes from the community surveillance video (540 frames), 14 keyframes from the supermarket surveillance video (450 frames), and 16 keyframes from the road surveillance video (510 frames). Our keyframe selection algorithm can filter a large number of redundant information frames generated by static pictures. Figure 6 shows some of the keyframes and key objects we selected from the scenes of community, supermarket, and road.

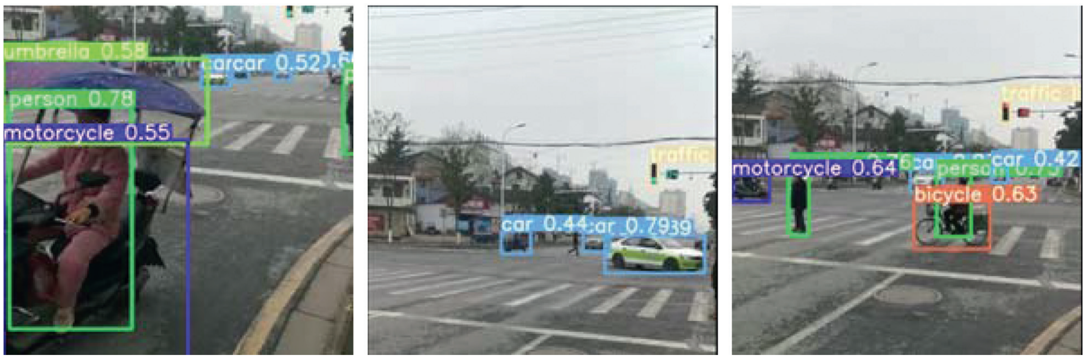
In the community scenario, the main monitoring object of the surveillance video is the ground parking lot. When there are no pedestrians and other objects on the ground parking lot and surrounding roads, the surveillance video is in a static image so no keyframe is extracted. As shown in Figure 6, when there are pedestrian objects in the community parking lot, the classes of surveillance video objects increase and the keyframes of the classes are extracted. When the pedestrian object is getting closer to the camera, the object detection frame and confidence become larger, and the confidence keyframe is extracted. When the number of pedestrians changes, the number keyframe is extracted. According to the extracted keyframes, we further extract key objects by setting the confidence threshold for the following super-resolution reconstruction. In the supermarket scene, the surveillance video mainly monitors the number of consumers before the product, and the frame that records the change in the number of consumers is the keyframe. By



Supermarket surveillance video detection objects



Community surveillance video detection objects



Road surveillance video detection objects

Figure 5: Object detection results of supermarkets, communities, and roads.

setting the confidence threshold, we can monitor and record the change frame of the consumer before a certain valuable product in real time. In road scenes, surveillance video mainly monitors vehicles and pedestrians, and frames that record the change in the number, type, and confidence of vehicles and pedestrians are keyframes.

5.4. Keyframe Object SR Experimental Results. Surveillance videos suffer from low resolution due to hardware technology, hardware cost, shooting environment, network transmission, and so on. To observe the details of key information, police and security personnel often need to magnify the image 4 times, 8 times, or even higher. If you directly magnify an image, the magnified image will miss a lot of high-frequency information and suffer from low quality and poor visual perception, making it difficult to recognize. The pixel-space-based super-resolution method can reconstruct SR video frames close to the real image by optimizing the distance between the super-resolved image and the real image in the pixel space. However, the reconstructed frames may lack a lot of high-frequency detail information, and the visual perception of texture features is unclear. Therefore, on the basis of super-resolution method based on pixel space, we further integrate the contextual loss based on feature space. By optimizing the local features of the super-resolved image and the real image in the feature space, we can improve the high-frequency information of the reconstructed video frames and obtain high-quality SR images with clear visual perception. It should be noted that, on the basis of object detection and keyframe selection, we select key objects from the keyframes according to the confidence threshold. The resolution and storage capacity are often very small and lack high-frequency information.

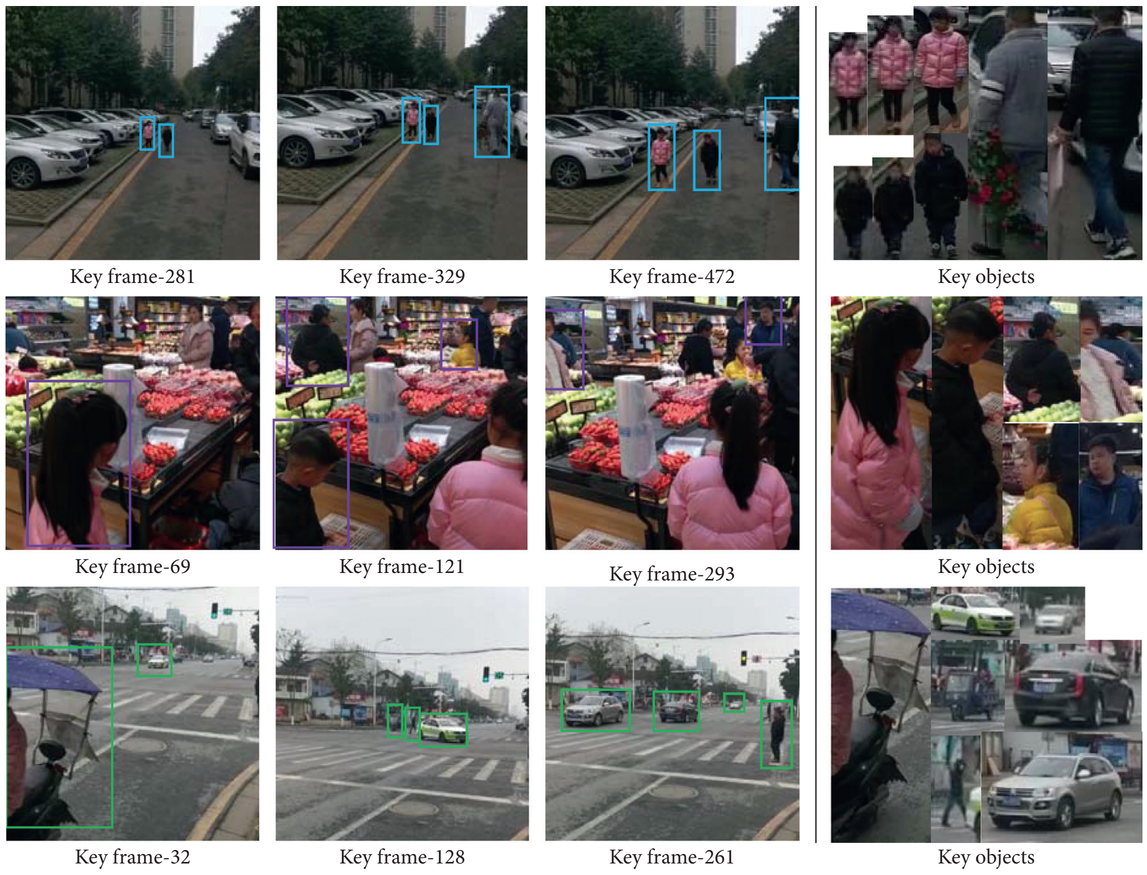


Figure 6: Some keyframes and key objects selected by the keyframe selection algorithm.

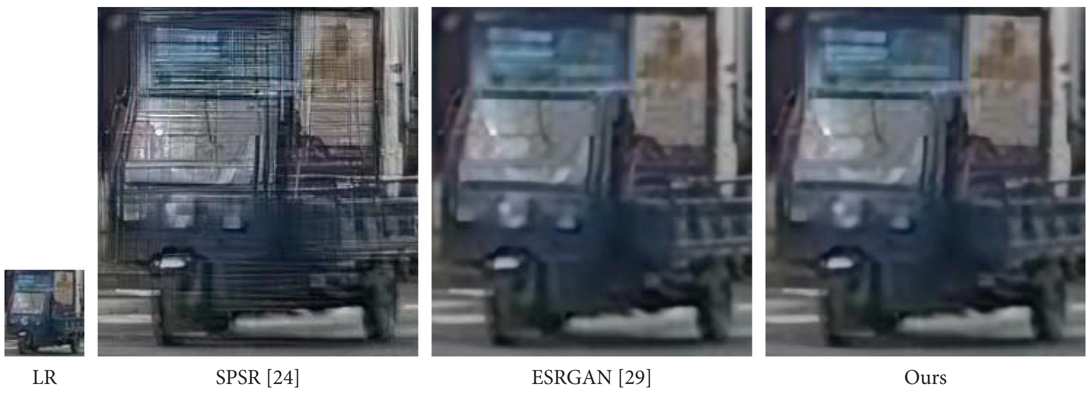


Figure 7: Comparison of super-resolved key objects (tricycles) in road surveillance video.

ESRGAN is the champion solution of the PIRM2018-SR (region 3) super-resolution competition, which can generate natural and detailed SR images. SPSR adds gradient map branches based on ESRGAN to constrain the structure of the generated image, which improves the authenticity of the SR image and the quality of visual perception. Therefore, the SPSR and ESRGAN [29], which can generate rich high-frequency information, were selected as our comparison methods. Figures 7–10 show the SR reconstruction results of the key objects in the three scenarios of roads, communities, and supermarkets.

We designed an asymmetric deep recursive back-projection network. We first constructed a back-projection structure in which the upper and lower sampling units are stacked in sequence by simulating the human visual system. Then, we used the cascading projection error and homologous projection error to correct the loss of the upsampling and downsampling units. In addition, we improved the SR reconstruction effect by cascading the outputs of upsampling units of different depths. We designed a recursive loop and asymmetric structure to improve the SR reconstruction effect without increasing the parameters. In the road scene, we choose a tricycle and a taxi

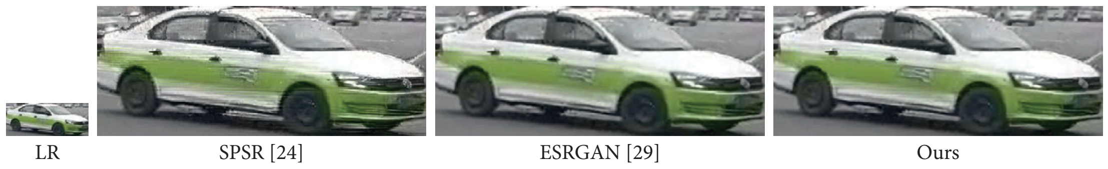


Figure 8: Comparison of super-resolved key objects (taxies) in road surveillance video.

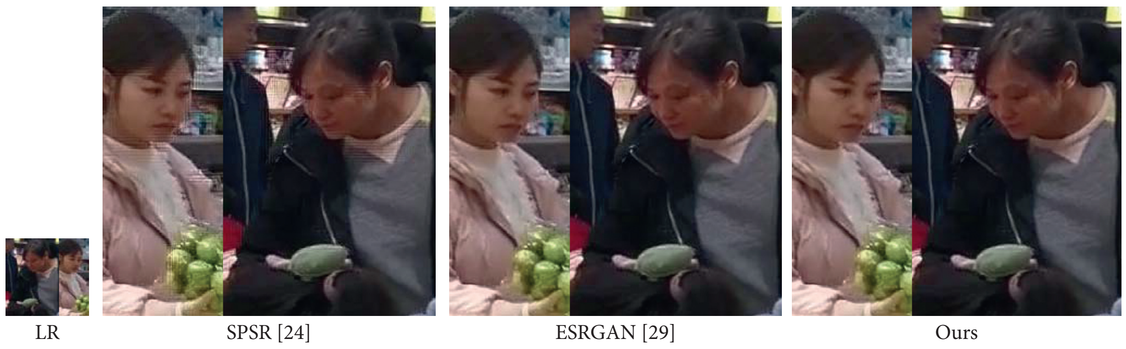


Figure 9: Comparison of super-resolved key objects (customers) in supermarket surveillance video.

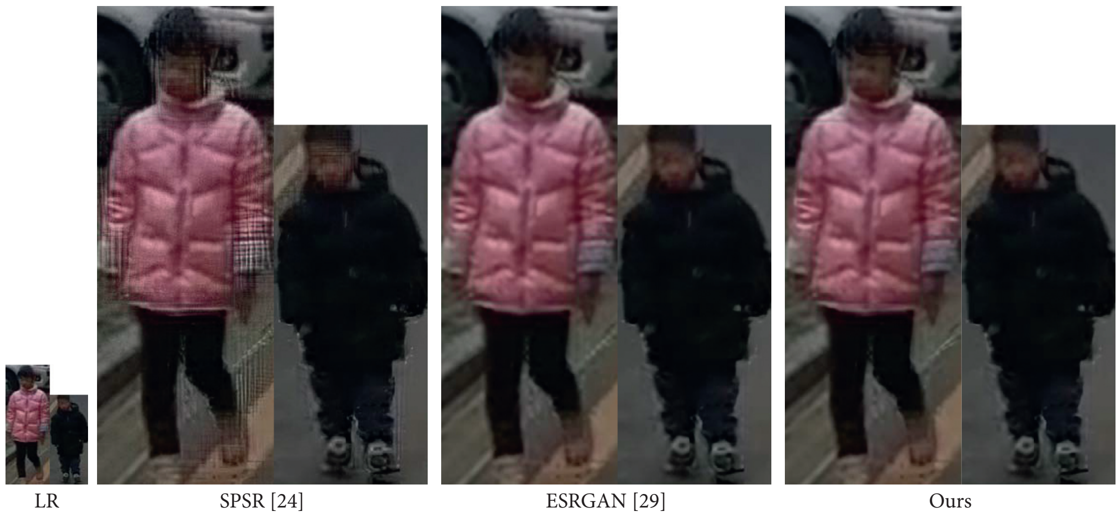


Figure 10: Comparison of super-resolved key objects (children) in community surveillance video.

for reconstruction and make comparison with the SPSR and ESRGAN. From Figure 7, we can see that the SPSR method reconstructs the SR image and produces a lot of artifacts, and many lines do not exist in the LR. The effect of ESRGAN reconstruction is typically similar to our method, but the edge part of the SR video frame reconstructed by our method is sharper and the visual perception is clearer. The effect of the reconstructed taxi in Figure 8 is the same as that of Figure 7. The SR video frame reconstructed by SPSR has artifacts on the underside of the taxi. Our method not only is free of artifacts but also has better clarity. In the community and supermarket scenes, we selected a girl, a boy, and two customers in the supermarket as the key objects for SR reconstruction. In Figure 10, we can see that the SPSR reconstructs artifacts in many places, such as the little girl's face and arms. When using real-world video frames for reconstruction, SPSR lacks generalization ability and robustness, while ESRGAN suffers from a mass of parameters, difficult model training, and insufficient visual perception quality. Our model uses the least parameters and achieves the best reconstruction effect.

6. Conclusion

Maintaining and ensuring the safety of the public domain constitute the foundation of safe and smart cities. Surveillance equipment widely distributed in the public domain can detect suspicious signs and objects in time, can inquire information about suspected vehicles and personnel through surveillance video, and can provide objective litigation evidence during the investigation and interrogation stage of the case. Traditional video surveillance systems rely on manual analysis of video content, which makes it difficult to effectively play the role of video surveillance. Therefore, we combine the traditional video surveillance system with artificial intelligence technology. Firstly, we fuse the object detection algorithm, keyframe selection algorithm, and super-resolution reconstruction algorithm to construct a super-resolution framework for video detection objects, which provides an effective auxiliary tool for the police personnel. Then, we employ the object detection algorithm to detect the object of the surveillance video in real time, which assists the police and other personnel to track the suspicious object. Besides, it helps select the type, quantity, and confidence of the surveillance video frames that have changed from a large amount of redundant information. Selecting key information from a large amount of redundant information can improve the efficiency of video analysis and reduce manual workload. Finally, to solve the problem that low-resolution surveillance video cannot be used effectively, we select the key object from the keyframe for super-resolution reconstruction. Combining the advantages of the pixel-based super-resolution method and the feature space loss function, we designed an asymmetric deep recursive back-projection network that can reconstruct the key objects with high resolution. The next step of our work is to realize super-resolution reconstruction of video detection objects under noise, blur, and other interference in the surveillance video.

Data Availability

The video, keyframe and key object images, and SR results data used to support the findings of this study have been deposited in the GitHub repository (https://github.com/yunfeiyoda/Video-Detection-Object-Super-resolution-.git).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors would like to thank the Hunan University of Arts and Science for providing them with high-performance servers. All the training and testing of the super-resolution model of the video detection object were completed on a high-performance server. This work was supported by the National Social Science Fund of China (no. 20&ZD120).

References

[1] K. Guo, B. Hu, J. Ma et al., "Toward anomaly behavior detection as an edge network service using a dual-task interactive guided neural network," *IEEE Internet of Things Journal*, vol. 99, 2020.

[2] L. Fang, Y. Li, X. Yun et al., "THP: a novel authentication scheme to prevent multiple attacks in SDN-based IoT network," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5745–5759, 2019.

[3] T. Mita, T. Kaneko, and O. Hori, "Joint haar-like features for face detection," in *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05)*, vol. 2, pp. 1619–1626, IEEE, Beijing, China, December 2005.

[4] L. Cuimei, Q. Zhiliang, J. Nan et al., "Human face detection algorithm via haar cascade classifier combined with three additional classifiers," in *Proceedings of the 13th IEEE International Conference on Electronic Measurement & Instruments (ICEMI)*, pp. 483–487, IEEE, Yangzhou, China, October 2017.

[5] S. Ren, K. He, R. Girshick et al., "Faster r-cnn: towards real-time object detection with region proposal networks," 2015, http://arXiv.org/abs/1506.01497.

[6] J. Redmon and A. Farhadi, "Yolov3: an incremental improvement," 2018, http://arXiv.org/abs/1804.02767.

[7] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

[8] C. Dong, C. C. Loy, K. He et al., "Image super-resolution using deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[9] C. Ledig, L. Theis, F. Huszár et al., "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4681–4690, Honolulu, HI, USA, July 2017.

[10] K. Guo, N. Li, J. Kang et al., "Towards efficient federated learning-based scheme in medical cyber-physical systems for distributed data," *Software: Practice and Experience*, 2020.

[11] R. Girshick, J. Donahue, T. Darrell et al., "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, IEEE, Columbus, OH, USA, June 2014.

[12] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448, IEEE, Santiago, Chile, December 2015.

[13] K. Guo, Y. Wang, J. Kang et al., "Core dataset extraction from unlabeled medical big data for lesion localization," *Big Data Research*, vol. 24, Article ID 100185, 2021.

[14] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "Yolov4: optimal speed and accuracy of object detection," 2020, http://arXiv.org/abs/2004.10934.

[15] W. Liu, D. Anguelov, D. Erhan et al., "Ssd: single shot multibox detector," *European Conference on Computer Vision*, Springer, Berlin, Germany, pp. 21–37, 2016.

[16] K. He, X. Zhang, S. Ren et al., "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, IEEE, Las Vegas, NV, USA, June 2016.

[17] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1646–1654, IEEE, Las Vegas, NV, USA, June 2016.

[18] X. Hu, H. Mu, X. Zhang et al., "Meta-SR: a magnification-arbitrary network for super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1575–1584, IEEE, Long Beach, CA USA, June 2019.

[19] K. Zhang, W. Zuo, and L. Zhang, "Deep plug-and-play super-resolution for arbitrary blur kernels," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1671–1681, IEEE, Long Beach, CA USA, June 2019.

[20] X. Xu, Y. Ma, and W. Sun, "Towards real scene super-resolution with raw images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1723–1731, IEEE, Long Beach, CA, USA, June 2019.

[21] Z. Li, J. Yang, Z. Liu et al., "Feedback network for image super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3867–3876, Long Beach, CA, USA, June 2019.

[22] S. Li, F. He, B. Du et al., "Fast spatio-temporal residual network for video super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10522–10531, IEEE, Long Beach, CA, USA, June 2019.

[23] J. Gu, H. Lu, W. Zuo et al., "Blind super-resolution with iterative kernel correction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1604–1613, IEEE, Long Beach, CA, USA, June 2019.

[24] T. Dai, J. Cai, Y. Zhang et al., "Second-order attention network for single image super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11065–11074, IEEE, Long Beach, CA, USA, June 2019.

[25] C. Ma, Y. Rao, Y. Cheng et al., "Structure-preserving super resolution with gradient guidance," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7769–7778, IEEE, Seattle, WA, USA, June 2020.

[26] Y. S. Xu, S. Y. R. Tseng, Y. Tseng et al., "Unified dynamic convolutional network for super-resolution with variational degradations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12496–12505, IEEE, Los Alamitos, CA, USA, June 2020.

[27] K. Guo, H. Guo, S. Ren, J. Zhang, and X. Li, "Towards efficient motion-blurred public security video super-resolution based on back-projection networks," *Journal of Network and Computer Applications*, vol. 166, p. 102691, 2020.

[28] S. Ren, J. Li, K. Guo, and F. Li, "Medical video super-resolution based on asymmetric back-projection network with multilevel error feedback," *IEEE Access*, vol. 9, pp. 17909–17920, 2021.

[29] X. Wang, K. Yu, S. Wu et al., "Esrgan: enhanced super-resolution generative adversarial networks," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pp. 1–16, Glasgow, UK, August 2018.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 9948808, 13 pages
https://doi.org/10.1155/2021/9948808

Review Article

Fabric Defect Detection in Textile Manufacturing: A Survey of the State of the Art

Chao Li,[1] Jun Li,[2] Yafei Li,[3] Lingmin He,[4] Xiaokang Fu,[5] and Jingjing Chen[1,6]

[1]*Zhijiang College, Zhejiang University of Technology, Hangzhou, China*
[2]*Binjiang College, Nanjing University of Information Science and Technology, Nanjing, China*
[3]*School of Information Engineering, Zhengzhou University, Zhengzhou, China*
[4]*College of Information Engineering, China Jiliang University, Hangzhou, China*
[5]*School of E-commerce & Management Science, Zhejiang Gongshang University, Hangzhou, China*
[6]*School of Economics, Fudan University, Shanghai, China*

Correspondence should be addressed to Jingjing Chen; joyjchan@gmail.com

Received 20 March 2021; Accepted 29 April 2021; Published 10 May 2021

Academic Editor: Xiaokang Zhou

Defects in the textile manufacturing process lead to a great waste of resources and further affect the quality of textile products. Automated quality guarantee of textile fabric materials is one of the most important and demanding computer vision tasks in textile smart manufacturing. This survey presents a thorough overview of algorithms for fabric defect detection. First, this review briefly introduces the importance and inevitability of fabric defect detection towards the era of manufacturing of artificial intelligence. Second, defect detection methods are categorized into traditional algorithms and learning-based algorithms, and traditional algorithms are further categorized into statistical, structural, spectral, and model-based algorithms. The learning-based algorithms are further divided into conventional machine learning algorithms and deep learning algorithms which are very popular recently. A systematic literature review on these methods is present. Thirdly, the deployments of fabric defect detection algorithms are discussed in this study. This paper provides a reference for researchers and engineers on fabric defect detection in textile manufacturing.

1. Introduction

The influence of artificial intelligence on industrial field has far exceeded our expectations [1, 2]. The vast number of researchers and engineers are constantly accelerating the development of industrial intelligence. In 2010, Germany put forward the Industry 4.0 framework, and the framework has been promoted and applied widely among the European Union member states. Subsequently, the United States and China put forward corresponding plans and policies for smart manufacturing of their own country. Industry 4.0 is the inevitable trend of the future development of manufacturing industry [3].

Industrial artificial intelligence is typical cross-disciplinary, which combines knowledge of mechanical, data science, network, communication, information security, and other disciplines, and it aims to use artificial intelligence algorithms to solve industrial problems and improve the efficiency and security of manufacturing [4, 5]. Towards industry 4.0, the textile manufacturing industry also needs to find its own way to adapt the manufacturing process. Textile manufacturing is a large-scale and complicated industry. The textile manufacturing process consists of a series of complex and orderly processes, mainly including spinning, weaving, dyeing, printing and finishing, and garments manufacturing. The stability and quality of the textile fabric produced by the whole production line are crucial to any enterprise [4].

There are many factors that affect the final product on the production line of textile manufacturing, such as material quality, mechanical factors, dye type, yarn size, and human factors [5]. In general, textile fabric defects refer to defects on the surface of the fabric. There are many types of fabric defects, most of which are caused by process problems and machine malfunctions. Defects will affect the quality of the final product, resulting in a great waste of all kinds of resources [6, 7]. In the process of fabric manufacturing, the defects in the previous stage will affect the later stage. Therefore, early detection of fabric defects can reduce the loss of enterprises earlier and faster [8]. Therefore, effective fabric defect detection is one of the key measures for modern fabric manufacturers to control cost and enhance product value and core competence.

In modern textile manufacturing, automatic fabric defect is an important way to ensure the textile quality [9]. For long, fabric defect detection is implemented by manual visual inspection which is inadequate and expensive in the meantime. Accordingly, automatic fabric defect detection is necessary for the textile industry to reduce cost and increase productivity. The core of a complete online textile fabric defect detection system is the detection algorithms. Many researchers and engineers in this field have devoted themselves to the design of robust and efficient algorithms within the past few decades [10]. Compared to manual fabric defects detection, the automatic detection systems are more effective with higher efficiency. Fabric defect detection has been a hot research field in computer science and technology and mechanical engineering. This paper provides not only the algorithms for the researchers but also the deployment problems for the practitioners [11].

This review summarizes and classifies the methods of fabric defect detection in a broader scope. A total of more than 2,000 articles were retrieved, from which 128 articles were included in this review. The main search terms used for retrieval are “Fabric defect detection,” “textile inspection,” “fabric defect recognition,” “automatic textile Manufacturing,” etc. In addition to the traditional and classical methods mentioned in the previous reviews, this review discusses and compares the algorithms of deep learning algorithms in defect detection which are very popular in recent years. Finally, the deployment of the algorithm is also discussed; for the algorithm, the deployment is also very important to the accuracy and efficiency of the system implementation. It is hoped that this review will provide some help and suggestions for the application of AI in fabric manufacturing.

2. Fabric Defect Detection Methods

Fabric defect detection algorithms are roughly divided into two categories in this study, traditional algorithms and learning-based algorithms, as shown in Figure 1. Most of the traditional algorithms are based on feature engineering with prior knowledge, covering statistical, structural, spectral, and model-based methods. The learning-based algorithms can be further divided into classical machine learning algorithms and deep learning algorithms. Machine learning uses mathematical algorithms to learn and analyze data to make predictions and take decisions in the future, which has been widely employed in recent years and achieved stratifying results in various disciplines and industries.

2.1. Traditional Algorithms

2.1.1. Statistical Algorithms. Statistical approaches utilize the spatial distribution of gray values in images [12], such as gray-level co-occurrence matrices (GLCM), autocorrelation analysis, and fractal dimension features.

Raheja et al. present an automated fabric defect detection system utilizing GLCM. In this approach, a signal graph is constructed with GLMC statistics and interpixel distance. Then a comparison between nondefective image and test image is made. Additionally, a Gabor filter based approach is utilized to detect the defects in this study. The conclusion is made that GLCM based algorithms generate higher detection accuracies and less computational complexity [13, 14].

Anandan et al. [15] combine the GLCM and curvelet transform (CT) by extracting the eigenvector of the defect which makes the fabric defect features more evident. The experiments show the effectiveness of the proposed algorithm with comparison to GLCM and wavelet-based methods, respectively.

Kumar et al. [16] design a statistical approach for identifying defects in fabric images using eigenvalues. Using the coefficient of variation, defective portions of the fabric images are identified. This method is simple and easy to use according to the experiments in the work.

Intending to effectively detect fabric defects, Song et al. [17] calculate the membership degree of each fabric region. Utilizing the extreme point density map of the image combined with the features of the membership function region, the saliency of defect regions is obtained. The whole scheme further adopts a threshold method and morphological processing. The author states this algorithm can detect fabric defects efficiently and accurately in the presence of noise and background texture interference.

Gharsallah et al. [18] present a fabric defect detection approach utilizing an improved anisotropic diffusion filter and saliency image features. Given that the conventional anisotropic diffusion methods cannot identify the defect edge which is confused with the background texture, the improved anisotropic diffusion method combines the local gradient magnitude with a saliency map. This approach can effectively remove the texture background and retain the image defect edge.

Table 1lists several statistical algorithms used for textile fabric defect detection in brief.

2.1.2. Spectral Approach. Fourier transform, Gabor transform, wavelet transform, and discrete cosine transform [22–24] are the representative method of spectral methods. These algorithms mentioned in this survey are listed in Table 2. Fourier transform, wavelet transform, and Gabor transform-based methods have been thoroughly studied and tested on fabric defect detection applications.

Li et al. [32] propose an algorithm employing a multi-scale wavelet transform and Gaussian mixture model for automated fabric defect detection. A textile fabric image was decomposed by the “Pyramid” wavelet transform and then reconstructed using thresholding method. Next, the

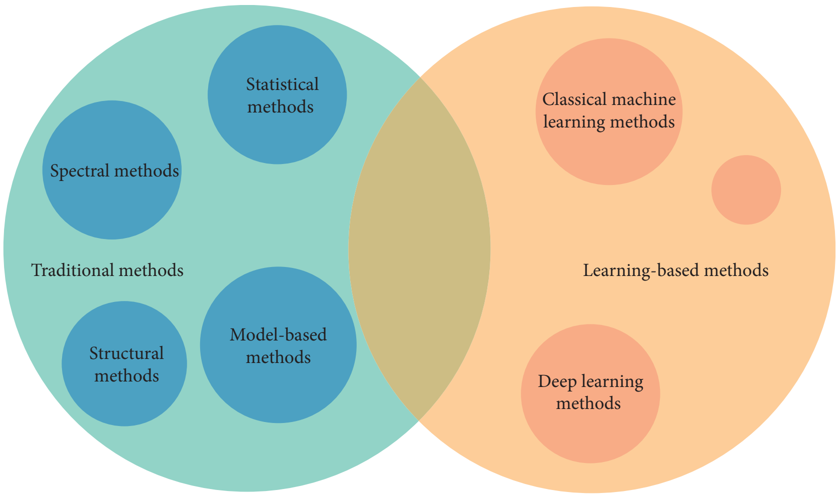


Figure 1: An overview of the defect detection methods covered in this review.

Table 1: Statistical algorithms for fabric defect detection.

Author	Proposed method	Dataset	Evaluation
Sayed [19]	Entropy filtering and minimum error thresholding	TILDA dataset	Detection success rate
Kumari [20]	Sylvester matrix based similarity method	KTH-TIPS-I and KTH-TIPS-II	False positives and false negatives
Chetverikov and Hanbury [21]	Based on two fundamental structural properties, regularity and local orientation (anisotropy)	Brodatz images and TILDA dataset	Detection success rate

Gaussian mixture model was utilized to segment the reconstructed image. The experiments demonstrate the effectiveness of the proposed algorithm for detecting and segmenting the defect images.

Rebhi et al. [33] present a fabric defect detection approach using local homogeneity information and discrete cosine transform (DCT). DCT was applied to the newly calculated homogeneity image and different energy features of all DCT blocks are then extracted. And the extracted features are fed into the feedforward neural networks classifier.

2.1.3. Structural Approach. One effective way for segmenting defective area on the patterned textile fabric image is the golden image subtraction (GIS) method. Ngan et al. proposed a method named wavelet preprocessed golden image subtraction (WGIS) [34]. Additionally, wavelet transforms, GIS, and WGIS methods are compared in this study, and the proposed WGIS achieved the best performance among them.

Jia and Liang [35] segment the fabric images into nonoverlapping regions named lattices and then the similarity of these lattices are calculated in the feature space. The proposed Isotropic Lattice Segmentation (ILS) method shows satisfying results on the box and star pattern image database. Jia et al. brought up another approach in their later study [36] on the basis of lattice segmentation and lattice templates. In this study, the lattices are segmented according to different placement rules of texture primitives that belong to different classes. The distances of undetermined lattice and lattice template are calculated, and the lattices are regarded as the defective area when the distances are larger than a certain threshold. The algorithm is further improved by adding template statistics which are learned from defect-free images in [37].

Another template-based correction approach for fabric images with periodical structure is introduced in Chang's work [38]. A fabric image is divided into lattices due to variation regularity and correction is then made to reduce the lattice misalignment. The defective lattices are first located and defect regions are segmented at a later step. Based on its assumptions, the lattice segmentation and template-based correction algorithms are more suitable for patterned fabric images.

Shi et al. [39] propose a method using low-rank decomposition of gradient information combined with structured graph. The fabric image is first divided into defect-free regions and defect regions based on the structured graph information. Adaptive thresholding is utilized during the lattice merging step. Finally, the matrix decomposition is calculated under the prior information from the segmentation results; thus the defect regions are emphasized. The presented method outperforms other methods on a standard database.

Abouelela et al. [40] design a fabric defect detection system employing simple statistical features such as median, mean, and variance. The author holds that time efficiency is crucial to any industrial procedure. Therefore, the author exchanged the complexity of simple features calculation for real-time performance. The proposed algorithm is better at detecting defects that vary drastically in the physical dimension.

Table 2: Spectral algorithms for fabric defect detection.

Author	Proposed method	Dataset	Evaluation
Sulochan [25]	Multiscale wavelet features and fuzzy C-means clustering	Real and computer-simulated fabric images	Detection error rate
Vermaak et al. [26]	Dual-tree complex wavelet transform (DTCWT)	TILDA dataset	Detection success rate
Liu and Zheng [27]	The method based on information entropy and frequency domain saliency	Database created by the research associate of the industrial automation research laboratory	ACC, true positive rate(TPR), false positive rate(FPR), positive predictive value (PPV), negative predictive value (NPV), time, F-measure
Di et al. [28]	L0 gradient minimization method and two-dimensional fractional Fourier transform (2D-FRFT) for obtaining the saliency map of the quaternion image	Dataset from automation laboratory fabric database of Hong Kong University	True positive (TP), false positive (FP), true negative (TN), and false negative (FN)
Jing [29]	Gabor preprocessed golden image subtraction	Industrial automation laboratory at the University of Hong Kong and the TILD database	Detection success rate
Mohammed and Alhamdani [30]	Fuzzy back propagation neural network (FBPNN) with Gabor features	Collected dataset	Detection success rate
Yapi et al. [31]	Using learning-based local textural distributions in the contourlet domain	TILDA database	(TP, FP, TN, and FN) local precision (P_L), local recall (R_L), and local accuracy (ACC_L)

2.1.4. Model-Based Methods. Ngan et al. propose motif-based methods for detecting defects in 2D patterned texture. This kind of methods is based on the assumption that patterned images can be divided into lattices and motifs. And further energy of moving subtraction is calculated to differentiate defective and defect-free region [41]. In order to reduce the detection rate of false positives and false negatives, the Gaussian mixture model is used to represent the energy variance value [42]. K-means clustering is applied to the data and the convex hull of each cluster is calculated in that fitting ellipsoidal region.

Lucia et al. propose an algorithm for detecting the fabric defects in uniformly structured textile fabric images. The algorithm includes two stages: the feature extraction stage and the defect recognition stage. In the first stage, the symmetric Gabor filter bank and principal component analysis are utilized for feature extraction, and in the second stage, the Euclidean norm of features is calculated and compared for defect recognition. This algorithm is designed on a patch basis and proved to be effective on the public TILDA database [43].

For environment-friendly textiles, Shu et al. [44] adopt an algorithm based on principal component analysis and nonlocal average filtering to enhance the fabric texture and reduce the noise. A texture-based defect measurement method is used for calculating the similarity; thus defect and nondefect areas can be distinguished.

Some researchers treat the fabric defect detection problem as a one-class classification. Bu et al. [45, 46] propose a method based on the support vector data description (SVDD) model. In the training stage, the multiple fractal features are extracted and the optimal parameters are selected for the Gaussian kernel function. The detection results on several datasets demonstrate that this combination is effective. Bu et al. [47] also present another method using SVDD model, and the features extracted in this method are based on autoregressive (AR) spectral estimation model combined with Burg algorithm.

With analysis on statistical model-based methods, Campbell et al. [48] propose two model-based methods in their study. The first method states the maximum likelihood of image binarization. The other method is mainly for defect detection in repeated weaving patterns; thus the discrete Fourier transformation is utilized for texture analysis. In the end, a model-based clustering method is applied to delineate the defective regions.

For the detection of patterned fabrics, Tsang et al. [49] propose a method named Elo rating (ER) in line with the spirit of sports. The fabric images are divided into standard-size partitions. Matches are calculated between partitions and revised through an Elo point matrix. The defect area (partition) will win the competition as a powerful player. This presented method was tested on dot-, star-, and box-patterned fabrics database and obtained comparable results to the most advanced method.

2.2. Learning-Based Algorithms

2.2.1. Classical Machine Learning Algorithms

(1) Dictionary Learning-Based Algorithms. Many researchers have validated the effectiveness of dictionary learning-based algorithms dealing with textile fabric defect detection problems [50, 51]. The general steps of these algorithms: first a dictionary is learned from the training or test image, and then a fabric image without defects is reconstructed using the learned dictionary; thenceforth the detection is implemented by subtracting the reconstructed image from the test

image. Recently, many algorithms based on low-rank representation have been brought up in the application of fabric defect inspection. In order to solve the optimization problem of the objective function, many methods reduce the low-rank decomposition problem to the nuclear minimization (NNM) problem.

Li et al. propose an algorithm on the basis of biological vision modeling. The biological visual saliency is modeled by low-rank representation (LRR); thus the fabric image is decomposed into salient defect regions and defect-free backgrounds [52].

Li et al. [53] model a defect-free region as a low-rank structure and the defect region as a sparse structure. Thus a fabric image can be regarded as the sum of a low-rank matrix and a sparse matrix. For dimensionality reduction, instead of singular value decomposition (SVD) on the matrix of the original image, the presented method uses eigenvalue decomposition on blocked image matrix. Therefore, this method is easy to implement and works well given that the contrast of the fabric image is sufficient.

Table 3tabulates some other dictionary learning-based algorithms using low-rank decomposition.

Shi et al. [39] point out two shortcomings of the low-rank decomposition. One is that existing low-rank decomposition models barely detect the defect regions with high gradients. And the other shortcoming is that small defect area or complex area will be incorrectly segmented given the inaccuracy of prior information. To overcome these shortcomings, Shi et al. propose a low-rank decomposition method utilizing gradient information combined with structured graph algorithm. The proposed method outperforms others on the point, box, and star databases.

Traditional machine learning algorithms, such as KNN [63] and neural network [64], are widely used in fabric defection detection problems. And feature engineering is one of the major processes in the machine learning life cycle.

Mak et al. extracted [65] four novel fractal features and employed support vector data description (SVDD), which is a support vector machine learning algorithm used for one-class classification, for fabric defect detection in his work.

Zhang et al. propose an approach employing the radial basis function (RBF) network. Gaussian mixture model (GMM) is utilized to improve the accuracy of Gaussian RBF parameter estimation. The validity of the proposed method has been proved on multiple class datasets [66].

Tian and Li [67] propose an autoencoder-based method for fabric defect detection by exploring similarities between image patches. Utilizing the repeated texture pattern, similar nondefective patches were found for each candidate defect patch and the corresponding latent variables were weighted and combined according to which the original latent variable can be modified. Experimental results manifest the effectiveness of the proposed algorithm.

Yapi et al. [68] consider this problem as a binary problem. A compact and accurate feature set was extracted by statistical modeling of multiscale contourlet decomposition, and then a Bayesian classifier (BC) is used to classify the defect and nondefect classes. This algorithm obtained high precision detection with real-time efficiency.

Some other traditional machine learning algorithms for fabric defect detection are shown in Table 4.

2.2.2. Deep Learning Algorithms. Recently, many researchers have applied deep learning techniques to fabric defect detection problems and have achieved satisfying results [72, 73] for the improvement of textile product quality and production efficiency [74]. Although deep learning methods have been proved to be powerful when dealing with segmentation and classification problems, there are still some problems in the practical application of specific industries [75]. First of all, the actual textile production line requires high real-time performance of the algorithm, which is the demand of high execution efficiency. Furthermore, compared to normal defect-free samples, the defective image data is difficult to obtain, which brings challenges to the training process of deep learning [76].

At present, the deep learning-based object detector can be classified as one-stage detectors and two-stage detectors [77]. Classical deep learning algorithms for object detection are listed in Table 5. In general, one-stage detectors have fast detection speed to meet the requirements of online detection, but the detection accuracy usually fails to meet requirements. In contrast, the two-stage algorithms have higher detection accuracy, but its detection speed is difficult to meet the real-time requirements of the algorithm in production scenes. In the field of fabric defect detection, the advantages and disadvantages of one-stage and two-stage detection algorithms are quite similar to those in other fields. The two-stage algorithm has higher accuracy but slower speed than the one-stage algorithm. In the actual application of textile industry, we hope that, under the premise of satisfying the detection accuracy, the faster the detection speed, the better. Therefore, the algorithm should be selected according to the actual application scenarios and requirements to find the balance between efficiency and accuracy.

(1) One-Stage Detection Algorithms. One-stage detection algorithm does not have a separate proposal generation phase. Typically, these algorithms treat all locations on the image as potential objects and manage to categorize each interest region into a target object or background.

The recently proposed single shot multibox detector (SSD) is a typical one-stage detector that has obtained good detection performance in object detection. This algorithm is designed based on a convolutional neural network (CNN). Some improvements have been made for the fabrics defect scenario by Liu et al. [78] and the experimental results show rationality and effectiveness.

Ouyang et al. [79] present a CNN based algorithm for on-loom fabric defect inspection. This proposed algorithm introduces a dynamic activation layer utilizing the defect probability information with a pairwise potential function to a CNN. This algorithm obtains good results dealing with the unbalanced data classification problem.

Deep convolutional neural network (DCNN) based algorithms have achieved satisfactory results on visual tasks and have been widely used in industrial scenarios. Liu et al. [80] employ DCNN to detect fabric defects with

Table 3: Dictionary learning algorithms for fabric defect detection.

Author	Proposed method	Dataset	Evaluation
Li et al. [52]	Low-rank representation (LRR)	(1) TILDA fabric images dataset; (2) dataset from the research associate of industrial automation research laboratory	Precision and recall
Li et al. [53]	Low-rank representation	500 fabric images from the textile kind C1 of the TILDA database	(a) Sensitivity and specificity; (b) false alarm rate (FAR), missing rate (MR)
Gao et al. [54]	Gabor filter and tensor low-rank recovery	Dataset from the research associate of industrial automation research laboratory	Receiver operating characteristic curve (ROC)
Shi et al. [39, 47]	Low-rank decomposition with gradient information	Dataset from the research associate of industrial automation research laboratory	TPR, FPR, PPV, NPV
Liu et al. [55–57]	Multi-scale convolutional neural network and low-rank decomposition model	(1) TILDA fabric images dataset; (2) dataset from the research associate of industrial automation research laboratory	Means and standard deviations of average precisions, recalls, F-measure, and mean absolute error (MAE)
Mo et al. [58]	Weighted double-low-rank decomposition method (WDLRD) to treat the matrix singular values differently by assigning different weights	Database is from the research associate of industrial automation research laboratory, HKBU	Visual defect locating results, the metrics of false alarm, recall, precision, accuracy, and F-measure
Li et al. [59]	Low-rank decomposition of multichannel feature matrices	(1) TILDA fabric images dataset; (2) dataset from the research associate of industrial automation research laboratory	ROC curves and precision-recall (PR) curves
Yang et al. [60]	Sparse and dense mixed low-rank decomposition	Real-world samples of 512∗512 with 256-gray levels	Saliency map (qualitative)
Wang et al. [61]	A randomized low-rank and sparse matrix decomposition model named GoDec	Fabric image dataset collected by Dr. Henry Y. T. Ngan [62]	Precision, recall, and F-measure

Table 4: Traditional machine learning algorithms for fabric defect detection.

Author	Proposed method	Dataset	Evaluation
Wang et al. [63]	Multiview stereo vision (MVS) and bag-of-features (BOF), K-nearest neighbor (KNN) algorithm	Collected dataset	Detection success rate
Priyanka and Manish [69]	Artificial neural networks (ANN)	Collected dataset	Detection success rate
Bumrungkun [70]	Snake active contour and support vector machines	Collected dataset	Recognition accuracy detection success rate
Zhang et al. [71]	L0 gradient minimization (LGM) and the fuzzy c-means (FCM) method to detect various fabric defects with diverse textures	Images from the automation laboratory sample database of Hong Kong University, TILDA textile texture database, and Guang Dong Esquel Textiles	ACC, TPR, FPR, PPV, and IOU (intersection over union)

complicated textures. This proposed approach is particularly designed for real textile production environment with limited resources. A series of improvements have been done to make the detection more effective. Zhou et al. propose an efficient DCNN architecture focusing on the problem of fabric defect detection, called Efficient Defect Detectors (EDDs) [81]. To extract more low-level features, EDDs adjust the input resolution, depth, and width using a scaling strategy. The improvement proved to be effective when compared with existing fabric defect detection algorithms.

Table 5: Deep learning algorithms for detecting object.

One-stage detectors	Two-stage detectors
YOLO	Faster RCNN
SSD	Mask RCNN
YOLOv2/v3/v4	Cascade RCNN
RefineDet	FPN
RetinaNet	R-FCN

Xu et al. [82] propose a novel detection network named de-deformation defect detection network (D4Net). This model is composed of reference generation, de-deformation network, and marginal loss. The most suitable reference is selected and paired with the input image and then is sent into the de-deformation network. The dissimilarities are calculated and enhanced by the marginal loss. Experiments on a large industrial database containing 67K images have been done and the results show that the algorithm outperforms other algorithms especially for large pattern fabric images.

Peng et al. put forward a detection algorithm called Priori Anchor Convolutional Neural Network (PRAN-Net) to fix this problem. Feature Pyramid Network (FPN) is utilized to selected multiscale feature maps and then sparse priori anchors are generated based on ground truth boxes [83].

Li et al. [84] propose an architecture using several microarchitectures. The microarchitecture is constructed of multiscale analysis, filter factorization, multilocation pooling, and parameters reduction, thus making the network a compact one. With the small model size, the proposed net worked well on fabric defect detection.

(2) Two-Stage Detection Algorithms. With regard to two-stage detectors, a sparse set of proposals is generated in the first stage and in the second stage, and the features of generated proposals are sent into DCNN for prediction results. As a remarkable two-stage detector, Faster R-CNN is an object detection model that improves on Fast R-CNN by utilizing a region proposal network (RPN) with the CNN model. Some researchers utilize the modified Faster R-CNN model for fabric defect detection [85–87]. Jun et al. [88] propose a framework that utilized the Inception-V1 model and LeNet-5 model. This approach includes local defect prediction in the first stage and global defect recognition in the second stage.

Table 6 lists some other deep learning algorithms utilized in textile fabric defect applications.

In recent years, Generative Adversarial Networks (GANs) have attracted a lot of attention [97, 98]. GANs and related algorithms have been widely used in a range of computer vision and computer graphics applications, such as image synthesis and video generation. GAN based fabric defect detection algorithms can automatically adapt to different fabric textures by learning existing fabric defect samples [99]. Liu et al. design a deep semantic segmentation network to detect fabric defects. They train a multistage GAN model to synthesize reasonable defect samples from nondefect samples. The performance of the method was verified by comprehensive experiments on all sorts of typical fabric image samples.

Le et al. [100] utilize Wasserstein generative adversarial nets (WGANs) combined with transfer learning techniques and multimodel ensembling framework. The effectiveness of the proposed scheme is demonstrated on unbalanced and rare datasets of images with defects.

Inspired by biological visual perception mechanism, Zhao et al. [101] describe a CNN model based on visual long-short-term memory (VLSTM). Three types of features, visual perception features, visual short-term memory (VSTM) features, and visual long-term memory (VLTM) features, are extracted by stacked convolutional autoencoders, a shallow CNN, and nonlocal neural networks, respectively. Experiments have been done on three public datasets and results show that the proposed algorithm is comparable to state-of-the-art algorithms.

Traditional saliency detection models usually rely on hand-crafted features to capture the information of global context and local details. Researchers add the attention mechanism to deal with fabric defect detection problems. Wang et al. [102] propose a deep saliency detection model that incorporated self-attention mechanism into a CNN for fabric defect detection. Multiscale feature maps are generated from a fully convolutional network, and a self-attention module is used to coordinate the dependence between the features of different layers. The self-attention mechanism in this algorithm proved to be very effective with complex or blurred defects.

Some studies combine the traditional and deep learning methods. Wang et al. [103] extracted global deep features using CNN in combination with handcrafted low-level features, and nonconvex robust PCA regularized by nonconvex total variation are employed to data processing and noise reduction. Then a segmentation algorithm is used to segment the saliency map to obtain the defect area.

3. Application and Deployment

When it comes to the deployment phase, there will be a lot of engineering implementation problems [104, 105]. Realizing an intelligent textile system in the real textile manufacturing process covers many aspects, involving the Internet of Things (IoT) [106], cyber-physical systems (CPS) [107], and more [108, 109].

3.1. Hardware Selection of Detection System. The basic components of image acquisition system are very important for the deployment work and therefore the hardware selection is critical for subsequent detection work [110]. Hardware such as cameras, lens, lights, and frame grabber is an important factor. Different hardware corresponds to different subsequent algorithms. For instance, Yildiz et al. [111] present a new fabric defect detection method for using a thermal camera. Fabric images obtained by the thermal imaging camera have their own image characteristics. The algorithm can be designed utilizing thermal differences between defect and defect-free areas and to improve the detection accuracy and efficiency while reducing the cost.

Different from visual inspection systems, Fang et al. [112] introduce a tactile inspection system for fabric defect detection. The system design is mainly based on a visual tactile sensor, which consists of several LEDs, a camera, and an elastic sensing layer. This system captures detailed information of surface structure neglecting of color and pattern; thus the algorithm designed in this study is mainly based on the structural information obtained from the tactile sensor.

In addition, the conveyor belt used for conveying cloth on the production line will also affect the image taking speed [113]. Therefore, the hardware selection of the entire system needs to be considered as a whole.

Table 6: Deep learning-based algorithms for fabric defect detection.

Author	Proposed or tested model	Categorize	Dataset	Evaluation
Jing et al. [89]	Mobile-Unet	One-stage	Benchmark databases, the fabric images database (FID), and yarn dyed fabric images (YFI), in which all images are manually annotated segmentation	Pixel accuracy (PA) and IoU
Hong-wei hang [90]	YOLOV2	One-stage	Collected dataset (276)	IOU, recall, and precision
Young-Joo Han [91]	Stacked convolutional autoencoders	One-stage	Synthetic and collected dataset	Recall, precision, and F-score
Xinying He [92]	Adaptive method based on DenseNet-SSD	One-stage	Collected dataset (2072)	Calculate localization loss (loc) and confidence loss (conf)
Mohammed et al. [93]	A multilayer perceptron with a Levenberg–Marquardt (LM) algorithm	One-stage	Collected dataset (217)	Specificity, accuracy, and sensitivity
Shuang mei [94]	Multiscale convolutional denoising autoencoder network model	One-stage	Four datasets: fabrics, KTH-TIPS, Kylberg texture, and ms-texture	Recall, precision, and F1-measure
Huosheng Xie [95]	Improved RefineDet	One-stage	TILDA dataset, Hong Kong patterned textures database, and DAGM2007 dataset	Precision (P), recall I, F1-score, mean average precision (mAP), model parameter (param.)
Yanqing Huang [96]	Segmentation network and decision network	Two-stage	Dark redfFabric (DRF), light blue fabric (LBF) and patterned texture fabric (PTF)	Frames per second (FPS) Avg-IoU and Avg-P

3.2. Dataset. This review lists a lot of learning-based algorithms. Although this type of method is very effective, it requires a large number of labeled fabric images with defects. However, it is very difficult to collect a fair amount of fabric defect image data in industrial scenes [114]. Therefore, many researchers employ semi-supervised and unsupervised learning algorithms for the detection [115]. In addition, some studies utilize non-defect image data and synthetic defective image data generated by using defect characteristics based on expert knowledge [91]. Chen et al. [116] propose a data augmentation method based on automatic image acquisition. Different image acquisition angles, various acquisition scenes, and random illumination conditions are designed for image collection as a simulation under the actual textile production scene.

In addition, the diversity of fabric defects and unbalanced categories bring actual challenges to fabric defect detection [86]. For instance, the multiscale defects in the fabric image will greatly increase the complexity and computation of the model; thus the designed fabric defect detection model of appropriate size must be able to meet the multiscale target detection.

3.3. Real Time of the Algorithm. Generally, the actual fabric defect detection task is implemented online on a platform with limited computing power. Thus the algorithms of online defect detection systems need to be accurate, efficient, and robust [117]. Therefore, the robustness and efficiency of the algorithm are critical to the actual production line. The computational cost of different algorithms is a critical consideration.

However, there are many other problems that exist in the textile production scenario. In order to obtain better image data and detection results, most existing algorithms require textile to be flattened. Therefore, some scholars design defect detection algorithms specifically for textiles with uneven and diverse shapes [118]. This consideration is closer to real-world settings.

Shunji et al. [119] design a detection method for tubular knitted fabric which is produced by a circular knitting machine. Vertical defects in circular knitted fabrics are caused by damaged needles. Once a vertical defect occurs during knitting, it will continue to exist unless the damaged needle is replaced with a new one.

In general, there is no real-time quality control system that can guarantee the quality in the production of noncrimp fabrics. The embedded system proposed by Schmitt et al. [120] ensures that all steps of image acquisition, processing, and evaluation can be executed in real time. The advantage of this proposed system is that real-time, accurate, and robust performance of the algorithm is ensured by detecting fiber orientation under industrial conditions.

In practical applications, fabric defect detection algorithms must not only ensure detection accuracy but also guarantee its applicability to hardware platforms with limited resources. Currently, the accuracy of existing detection models is low. This is due in large part to multiscale defects in the fabric image; so, the fabric defect detection model must be able to meet multiscale object detection. However, even the best model is still troubled by the large size of the problem. Therefore, we must consider ways to reduce the size of the model. Inspired by the successful use of deep convolutional neural networks (DCNN) for target detection, we propose a wide-and-light network structure called WALNet.

4. Discussion

Fabric defects correspond to defects on the surface of the textile fabric. Most fabric defects are caused by machine or process faults and malfunctions. The existence of fabric defects greatly reduces the sale and use of textiles. Textile manufacturing companies need to upgrade equipment and technology to maintain growth and competitiveness.

The sensing, storage, and computing capabilities of automated fabric detection systems based on computer vision will continue to improve. The development of hardware and algorithms will greatly affect the accuracy of detection and the ease of deployment.

Besides the fabric defect detection phase discussed in this survey, there is a lot of work that needs to be done during the whole textile manufacturing process. A lot of research works have been proposed for yarn production, fabric manufacturing, and finishing process utilizing learning-based methods. Huynh [121] proposes an online fabric defect prediction method based on the back propagation neural network models. The acquired data is collected in the form of time series and then converted into regional data based on the control chart. The proposed model can predict the defect types in advance and thus can reduce the workload of quality control in the production process.

In the future, more work needs to be done in the process of moving towards Industry 4.0 [122, 123]. Smart manufacturing integrates various technologies, covering robotics, CPS, IoT, big data analytics [124], and cloud computing. CPS is an engineering system that seamlessly integrates physical and computational components [125]. Adding artificial intelligence, big data analysis, and cloud services to the IoT ecosystem is the key development direction of CPS in the future.

5. Conclusions

This paper presents a systematic literature review on automatic fabric defect detection methods of the textile industry smart manufacturing. All the methods covered in this work are roughly classified into two main categories, namely, traditional algorithms and learning-based algorithms. There are no clear boundaries between the different categories. To realize a better detection result, the researchers often combine different algorithms. The research results of this survey also confirmed that better results can be obtained by combining different methods and thus provide suggestions and ideas for further research. Accurate, efficient, and robust fabric defect detection algorithms are necessary to develop fully automated web detection systems.

The automatic textile fabric defect detection technology based on computer vision has attracted great attention of researchers. With the development of new object detection algorithms, computational capabilities, and sensor technology and industry, computer-vision based textile defect detection techniques will continue to evolve at a high speed.

Conflicts of Interest

The authors declare no conflicts of interest.

Authors' Contributions

Research plan was carried out by Jingjing Chen and Chao Li; original draft preparation was performed by Chao Li; reviewing and editing were performed by Jingjing Chen, Jun Li, Yafei Li, Lingmin He, and Xiaokang Fu.

Acknowledgments

This research was funded by Zhejiang Provincial Basic Public Welfare Research Program, grant number LQ20F020024.

References

[1] J. Lee, *Industrial AI*, Springer Singapore, Singapore, 2020.

[2] K. Yan, L. Liu, Y. Xiang, and Q. Jin, "Guest editorial: AI and machine learning solution cyber intelligence technologies: new methodologies and applications," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6626–6631, 2020.

[3] K.-D. Thoben, S. Wiesner, S. Wiesner, and T. Wuest, ""Industrie 4.0" and smart manufacturing—a review of research issues and application examples," *International Journal of Automation Technology*, vol. 11, no. 1, pp. 4–16, 2017.

[4] A. Kumar, "Computer vision-based fabric defect analysis and measurement," in *Computer Technology for Textiles and Apparel: Woodhead Publishing Series in Textiles*, J. Hu, Ed., pp. 45–65, Woodhead Publishing, Cambridge, UK, 2011.

[5] X. Xu, D. Cao, Y. Zhou, and J. Gao, "Application of neural network algorithm in fault diagnosis of mechanical intelligence," *Mechanical Systems and Signal Processing*, vol. 141, p. 106625, 2020.

[6] J. Bullon, A. González Arrieta, A. Hernández Encinas et al., "Manufacturing processes in the textile industry. Expert Systems for fabrics production," *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, vol. 6, no. 4, pp. 15–23, 2017.

[7] M. Sibly Sadik, *Defects of Woven Fabrics and Their Remedies*, Bachelor of Science, Department of Textile Engineering, Daffodil International University, Dhaka, Bangladesh, 2014.

[8] K. Singh, J. Kaleka, and J. Kaleka, "Identification and classification of fabric defects," *International Journal of Advanced Research*, vol. 4, no. 8, pp. 1137–1141, 2016.

[9] L. Song, R. Li, and S. Chen, "Fabric defect detection based on membership degree of regions," *IEEE Access*, vol. 99, p. 1, 2020.

[10] P. M. Mahajan, S. R. Kolhe, and P. M. Patil, "A review of automatic fabric defect detection techniques," *Advances in Computational Research*, vol. 1, no. 2, pp. 18–29, 2009.

[11] H. Y. T. Ngan, G. K. H. Pang, and N. H. C. Yung, "Automated fabric defect detection-A review," *Image and Vision Computing*, vol. 29, no. 7, pp. 442–458, 2011.

[12] M. T. Habib, S. B. Shuvo, M. S. Uddin et al., "Automated textile defect classification by bayesian classifier based on statistical features," in *Proceedings of the 2016 International Workshop on Computational Intelligence (IWCI)*, Dhaka, Bangladesh, December 2016.

[13] J. L. Raheja, S. Kumar, and A. Chaudhary, "Fabric defect detection based on GLCM and Gabor filter: a comparison," *Optik*, vol. 124, no. 23, pp. 6469–6474, 2013.

[14] J. L. Raheja, B. Ajay, and A. Chaudhary, "Real time fabric defect detection system on an embedded DSP platform," *Optik*, vol. 124, no. 21, pp. 5280–5284, 2013.

[15] R. S. Sabeenian, "Fabric defect detection using discrete curvelet transform," *Procedia Computer Science*, vol. 133, pp. 1056–1065, 2018.

[16] P. Senthil Kumar and H. Hafedh, "Detection of defects in knitted fabric images using Eigen values," *International Journal Computer Science Engineering-IJASCSE*, vol. 2, no. 3, pp. 7–10, 2013.

[17] L. Song, R. Li, and S. Chen, "Fabric defect detection based on membership degree of regions," *IEEE Access*, vol. 99, p. 1, 2020.

[18] M. B. Gharsallah and E. B. Braiek, "A visual attention system based anisotropic diffusion method for an effective textile defect detection," *Journal of the Textile Institute*, vol. 8, pp. 1–15, 2020.

[19] M. S. Sayed, "Robust fabric defect detection algorithm using entropy filtering and minimum error thresholding," in *Proceedings of the 2016 IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Abu Dhabi, UAE, October 2016.

[20] R. M. L. N. Kumari, "Sylvester matrix based similarity estimation method for automation of defect detection in textile fabrics," *Journal of Sensors*, vol. 2021, Article ID 6625421, 11 pages, 2021.

[21] D. Chetverikov and A. Hanbury, "Finding defects in texture using regularity and local orientation," *Pattern Recognition*, vol. 35, no. 10, pp. 2165–2180, 2002.

[22] L. Chen, S. Zeng, Q. Gao et al., "Adaptive gabor filtering for fabric defect inspection," *Journal of Compurters*, vol. 31, no. 2, pp. 45–55, 2020.

[23] J. Zhang, Y. Li, and H. Luo, "Defect detection in textile fabrics with optimal Gabor filter and BRDPSO algorithm," *Journal of Physics: Conference Series*, vol. 1651, p. 012073, 2020.

[24] B. A. TanveerSajid, "Fabric defect detection in textile images using gabor filter," *IOSR Journal of Electrical and Electronics Engineering*, vol. 3, no. 2, 2012.

[25] H. C. Sulochan, "Fabric weave pattern detection based on fuzzy clustering and texture orientation features in wavelet domain," *Journal of Textile Science & Engineering*, vol. 8, no. 6, 2018.

[26] N. Vermaak, P. Nsengiyumva, and N. Luwes, "Using the dual-tree complex wavelet transform for improved fabric defect detection," *Journal of Sensors*, vol. 2016, Article ID 9794723, 8 pages, 2016.

[27] G. Liu and X. Zheng, "Fabric defect detection based on information entropy and frequency domain saliency," *Visual Computer*, vol. 24, 2020.

[28] L. Di, H. Long, and J. Liang, "Fabric defect detection based on illumination correction and visual salient features," *Sensors*, vol. 20, no. 18, 2020.

[29] J.-F. Jing, S. Chen, and P.-F. Li, "Fabric defect detection based on golden image subtraction," *Coloration Technology*, vol. 133, 2017.

[30] I. S. Mohammed and I. M. Alhamdani, "A fuzzy system for detection and classification of textile defects to ensure the quality of fabric production," *International Journal of Electrical and Computer Engineering*, vol. 9, no. 5, 2019.

[31] D. Yapi, M. S. Allili, and N. Baaziz, "Automatic fabric defect detection using learning-based local textural distributions in the contourlet domain," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 3, pp. 1014–1026, 2018.

[32] P. Li, H. Zhang, J. Jing, R. Li, and J. Zhao, "Fabric defect detection based on multi-scale wavelet transform and Gaussian mixture model method," *The Journal of The Textile Institute*, vol. 106, no. 6, pp. 587–592, 2015.

[33] A. Rebhi, I. Benmhammed, S. Abid, and F. Fnaiech, "Fabric defect detection using local homogeneity analysis and neural network," *Journal of Photonics*, vol. 2015, Article ID 376163, 9 pages, 2015.

[34] H. Y. T. Ngan, G. K. H. Pang, S. P. Yung, and M. K. Ng, "Wavelet based methods on patterned fabric defect detection," *Pattern Recognition*, vol. 38, no. 4, pp. 559–576, 2005.

[35] L. Jia and J. Liang, "Fabric defect inspection based on isotropic lattice segmentation," *Journal of the Franklin Institute*, vol. 354, no. 13, pp. 5694–5738, 2017.

[36] L. Jia, J. Zhang, S. Chen, and Z. Hou, "Fabric defect inspection based on lattice segmentation and lattice templates," *Journal of the Franklin Institute*, vol. 355, no. 15, pp. 7764–7798, 2018.

[37] L. Jia, C. Chen, S. Xu, and J. Shen, "Fabric defect inspection based on lattice segmentation and template statistics," *Information Sciences*, vol. 512, pp. 964–984, 2020.

[38] X. Chang, C. Gu, J. Liang et al., "Fabric defect detection based on pattern template correction," *Mathematical Problems in Engineering*, vol. 2018, Article ID 3709821, 17 pages, 2018.

[39] B. Shi, J. Liang, L. Di, C. Chen, and Z. Hou, "Fabric defect detection via low-rank decomposition with gradient information and structured graph algorithm," *Information Sciences*, vol. 546, pp. 608–626, 2021.

[40] A. Abouelela, H. M. Abbas, H. Eldeeb, A. A. Wahdan, and S. M. Nassar, "Automated vision system for localizing structural defects in textile fabrics," *Pattern Recognition Letters*, vol. 26, no. 10, pp. 1435–1443, 2005.

[41] H. Y. T. Ngan, G. K. H. Pang, and N. H. C. Yung, "Motif-based defect detection for patterned fabric," *Pattern Recognition*, vol. 41, no. 6, pp. 1878–1894, 2008.

[42] H. Y. T. Ngan, G. K. H. Pang, and N. H. C. Yung, "Ellipsoidal decision regions for motif-based patterned fabric defect detection," *Pattern Recognition*, vol. 43, no. 6, pp. 2132–2144, 2010.

[43] L. Bissi, G. Baruffa, P. Placidi, E. Ricci, A. Scorzoni, and P. Valigi, "Automated defect detection in uniform and structured fabrics using Gabor filters and PCA," *Journal of Visual Communication and Image Representation*, vol. 24, no. 7, pp. 838–845, 2013.

[44] Y. Shu, L. Zhang, D. Zuo et al., "Analysis of texture enhancement methods for the detection of eco-friendly textile fabric defects," *Journal of Intelligent & Fuzzy Systems*, pp. 1–11, 2021.

[45] H.-G. Bu, J. Wang, and X.-B. Huang, "Fabric defect detection based on multiple fractal features and support vector data description," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 2, pp. 224–235, 2009.

[46] H.-G. Bu, X.-B. Huang, J. Wang, and X. Chen, "Detection of fabric defects by auto-regressive spectral analysis and support vector data description," *Textile Research Journal*, vol. 80, no. 7, pp. 579–589, 2010.

[47] B. Shi, J. Liang, L. Di et al., "Fabric Defect Detection via Low-Rank Decomposition With Gradient Information," *IEEE Access*, vol. 99, p. 1, 2019.

[48] J. G. Campbell, C. Fraley, D. Stanford, F. Murtagh, and A. E. Raftery, "Model-based methods for textile fault detection," *International Journal of Imaging Systems and Technology*, vol. 10, no. 4, pp. 339–346, 1999.

[49] C. S. C. Tsang, H. Y. T. Ngan, and G. K. H. Pang, "Fabric inspection based on the Elo rating method," *Pattern Recognition*, vol. 51, pp. 378–394, 2016.
[50] L. Tong, W. K. Wong, and C. K. Kwong, "Fabric defect detection for apparel industry: a nonlocal sparse representation approach," *IEEE Access*, vol. 5, pp. 5947–5964, 2017.
[51] X. Kang and E. Zhang, "A universal and adaptive fabric defect detection algorithm based on sparse dictionary learning," *IEEE Access*, vol. 99, p. 1, 2020.
[52] C. Li, G. Gao, Z. Liu, M. Yu, and D. Huang, "Fabric defect detection based on biological vision modeling," *IEEE Access*, vol. 6, pp. 27659–27670, 2018.
[53] P. Li, J. Liang, X. Shen, M. Zhao, and L. Sui, "Textile fabric defect detection based on low-rank representation," *Multimedia Tools and Applications*, vol. 78, no. 1, pp. 99–124, 2019.
[54] G. Gao, C. Liu, Z. Liu et al., Eds., in *Proceedings of the 2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*, Nanjing, China, November 2017.
[55] Z. Liu, B. Wang, C. Li et al., "Fabric defect detection based on deep-feature and low-rank decomposition," *Journal of Engineered Fibers & Fabrics*, vol. 15, 2020.
[56] Z. Liu, B. Wang, C. Li et al., "Fabric defect detection algorithm based on convolution neural network and low-rank representation," in *Proceedings of the 2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*, Nanjing, China, November 2018.
[57] Z. Liu, L. Yan, C. Li, Y. Dong, and G. Gao, "Fabric defect detection based on sparse representation of main local binary pattern," *International Journal of Clothing Science and Technology*, vol. 29, no. 3, pp. 282–293, 2017.
[58] D. Mo, W. K. Wong, Z. Lai, and J. Zhou, "Weighted double-low-rank decomposition with application to fabric defect detection," *IEEE Transactions on Automation Science and Engineering*, vol. 99, pp. 1–21, 2020.
[59] C. Li, C. Liu, G. Gao, Z. Liu, and Y. Wang, "Robust low-rank decomposition of multi-channel feature matrices for fabric defect detection," *Multimedia Tools and Applications*, vol. 78, no. 6, pp. 7321–7339, 2019.
[60] Y. Yang, J. Wang, Z. Liu et al., "Fabric defect detection method based on sparse and dense mixed low-rank decomposition," in *Proceedings of the 2018 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, Shenzhen, China, December 2018.
[61] J. Wang, Q. Li, J. Gan et al., "Fabric defect detection based on improved low-rank and sparse matrix decomposition," in *Proceedings of the 2017 IEEE International Conference on Image Processing (ICIP)*, Beijing, China, September 2017.
[62] Y. Ngan, G. Pang, S. Yung et al., "Defect detection on patterned jacquard fabric," in *Proceedings of the 32nd Applied Image Pattern Recognition Workshop (AIPR 2003), Image Data Fusion*, Washington, DC, USA, October 2003.
[63] Y. Wang, N. Deng, and B. Xin, "Investigation of 3D surface profile reconstruction technology for automatic evaluation of fabric smoothness appearance," *Measurement*, vol. 166, p. 108264, 2020.
[64] S. Kulkarni, K. Jojare, V. Bhosale, and P. Arude, "Textile fabric defect detection," *IJARCCE*, vol. 5, no. 12, pp. 476–478, 2016.
[65] K. L. Mak, P. Peng, and K. F. C. Yiu, "Fabric defect detection using morphological filters," *Image and Vision Computing*, vol. 27, no. 10, pp. 1585–1592, 2009.
[66] Y. Zhang, Z. Lu, and J. Li, "Fabric defect classification using radial basis function network," *Pattern Recognition Letters*, vol. 31, no. 13, pp. 2033–2042, 2010.
[67] H. Tian and F. Li, "Autoencoder-based fabric defect detection with cross- patch similarity," in *Proceedings of the 2019 16th International Conference on Machine Vision Applications (MVA)*, Tokyo, Japan, May 2019.
[68] D. Yapi, M. Mejri, M. S. Allili, and N. Baaziz, "A learning-based approach for automatic defect detection in textile images," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 2423–2428, 2015.
[69] V. Priyanka and K. Manish, "Fabric defect inspection system using neural network," *International Journal of Multidisciplinary Research and Development*, vol. 2, no. 4, pp. 569–573, 2015.
[70] P. Bumrungkun, "Defect detection in textile fabrics with snake active contour and support vector machines," *Journal of Physics: Conference Series*, vol. 1195, Article ID 012006, 2019.
[71] H. Zhang, J. Ma, J. Jing, and P. Li, "Fabric defect detection using L0 gradient minimization and fuzzy C-means," *Applied Sciences*, vol. 9, no. 17, p. 3506, 2019.
[72] D. Siegmund, B. Fu, A. Jose-Garcia et al., "Study and research on detection of fiber defects using keypoints and deep learning," *International Journal of Pattern Recognition & Artificial Intelligence*, 2020.
[73] S. Das, A. Wahi, S. Keerthika, and N. Thulasiram, "Defect analysis of textiles using artificial neural network," *Current Trends in Fashion Technology & Textile Engineering*, vol. 6, no. 1, 2020.
[74] S. Barua, H. Patil, P. D. Desai et al., "Deep learning-based smart colored fabric defect detection system," *Applied Computer Vision and Image Processing*, pp. 212–219, 2020.
[75] X. Zhou, Y. Li, and W. Liang, "CNN-RNN based intelligent recommendation for online medical pre-diagnosis support," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, p. 1, 2020.
[76] Y. Li, W. Zhao, and J. Pan, "Deformable patterned fabric defect detection with Fisher criterion-based deep learning," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 1256–1264, 2017.
[77] X. Wu, D. Sahoo, and S. C. H. Hoi, "Recent advances in deep learning for object detection," *Neurocomputing*, vol. 396, pp. 39–64, 2020.
[78] Z. Liu, S. Liu, and C. Li, "Fabric defects detection based on SSD," in *Proceedings of the 2nd International Conference on Graphics and Signal Processing: ICGSP'18*, et al. , Sydney, Australia, October 2018.
[79] W. Ouyang, B. Xu, J. Hou, and X. Yuan, "Fabric defect detection using activation layer embedded convolutional neural network," *IEEE Access*, vol. 7, pp. 70130–70140, 2019.
[80] Z. Liu, C. Zhang, C. Li et al., "Fabric defect recognition using optimized neural networks," *Journal of Engineered Fibers and Fabrics*, vol. 14, 2019.
[81] T. Zhou, J. Zhang, H. Su, W. Zou, and B. Zhang, "EDDs: a series of Efficient Defect Detectors for fabric quality inspection," *Measurement*, vol. 172, p. 108885, 2021.
[82] X. Xu, J. Chen, H. Zhang, and W. W. Y. Ng, "D4Net: de-deformation defect detection network for non-rigid products with large patterns," *Information Sciences*, vol. 547, pp. 763–776, 2021.
[83] P. Peng, Y. Wang, C. Hao, Z. Zhu, T. Liu, and W. Zhou, "Automatic fabric defect detection method using PRAN-net," *Applied Sciences*, vol. 10, no. 23, p. 8434, 2020.
[84] Y. Li, D. Zhang, and D.-J. Lee, "Automatic fabric defect detection with a wide-and-compact network," *Neurocomputing*, vol. 329, pp. 329–338, 2019.

[85] H. Zhou, B. Jang, Y. Chen et al., "Exploring faster RCNN for fabric defect detection," in *Proceedings of the 2020 Third International Conference on Artificial Intelligence for Industries (AI4I)*, Irvine, CA, USA, September 2020.

[86] Z. Zhao, K. Gui, and P. Wang, "Fabric defect detection based on cascade faster R-CNN," in *Proceedings of the 4th International Conference on Computer Science and Application Engineering*, Sanya, China, October 2020.

[87] J. Wu, J. Le, Z. Xiao et al., "Automatic fabric defect detection using a wide-and-light network," *Applied Intelligence*, pp. 1–17, 2021.

[88] X. Jun, J. Wang, J. Zhou et al., "Fabric defect detection based on a deep convolutional neural network using a two-stage strategy," *Textile Research Journal*, vol. 91, no. 1-2, pp. 130–142, 2020.

[89] J. Jing, Z. Wang, M. Rtsch et al., "Mobile-Unet: an efficient convolutional neural network for fabric defect detection," *Textile Research Journal*, 2020.

[90] H. Zhang, L. Zhang, P. Li et al., "Yarn-dyed fabric defect detection with YOLOV2 based on deep convolution neural networks," in *Proceedings of the 2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS)*, pp. 170–174, Enshi, China, May 2018.

[91] Y.-J. Han and H.-J. Yu, "Fabric defect detection system using stacked convolutional denoising auto-encoders trained with synthetic defect data," *Applied Sciences*, vol. 10, no. 7, p. 2511, 2020.

[92] X. He, L. Wu, and F. Song, "Research on Fabric defect detection based on deep fusion DenseNet-SSD network," in *Proceedings of the IcWCSN 2020: 2020 International Conference on Wireless Communication and Sensor Networks*, et al. , Warsaw Poland, May 2020.

[93] K. M. C. Mohammed, "Defective texture classification using optimized neural network structure," *Pattern Recognition Letters*, vol. 135, pp. 228–236, 2020.

[94] S. Mei, Y. Wang, and G. Wen, "Automatic fabric defect detection with a multi-scale convolutional denoising autoencoder network model," *Sensors (Basel, Switzerland)*, vol. 18, no. 4, 2018.

[95] H. Xie and Z. Wu, "A robust fabric defect detection method based on improved RefineDet," *Sensors*, vol. 20, no. 15, p. 4260, 2020.

[96] Y. Huang, J. Jing, and Z. Wang, "Fabric defect segmentation method based on deep learning," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, 2021.

[97] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial nets,"vol. 2, pp. 2672–2680, in *Proceedings of the 27th International Conference on Neural Information Processing Systems*, vol. 2, pp. 2672–2680, MIT Press, Montreal, Canada, December 2014.

[98] K. Yan, A. Chong, and Y. Mo, "Generative adversarial network for fault detection diagnosis of chillers," *Building and Environment*, vol. 172, p. 106698, 2020.

[99] J. Liu, C. Wang, H. Su et al., "Multistage GAN for fabric defect detection," *IEEE Transactions on Image Processing*, vol. 29, pp. 3388–3400, 2020.

[100] X. Le, J. Mei, H. Zhang et al., "A learning-based approach for surface defect detection using small image datasets," *Neurocomputing*, vol. 408, pp. 112–120, 2020.

[101] Y. Zhao, K. Hao, H. He et al., "A visual long-short-term memory based integrated CNN model for fabric defect image classification," *Neurocomputing*, vol. 380, pp. 259–270, 2020.

[102] J. Wang, Z. Liu, C. Li et al., "Self-attention deep saliency network for fabric defect detection," *Communications in Computer and Information Science*, Springer, vol. 1160Singapore, , 2020.

[103] J. Wang, C. Li, Z. Liu et al., *Combing Deep and Handcrafted Features for NTV-NRPCA Based Fabric Defect Detection*, Springer, Cham, Switzerland, 2019.

[104] J. Lu, J. Wang, D. Chen et al., "A service-oriented tool-chain for model-based systems engineering of aero-engines," *IEEE Access*, vol. 6, pp. 50443–50458, 2018.

[105] J. Lu, G. Wang, and M. Törngren, "Design ontology in a case study for cosimulation in a model-based systems engineering tool-chain," *IEEE Systems Journal*, vol. 14, no. 1, pp. 1297–1308, 2020.

[106] X. Zhou, W. Liang, I. K. Wang et al., "Deep learning enhanced human activity recognition for Internet of healthcare Things," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6429–6438, 2020.

[107] J. Lu, D. Chen, G. Wang, D. Kiritsis, M. Tórngren et al., "Model-based systems engineering tool-chain for automated parameter value selection," *in IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.

[108] V. Özdemir, "The big picture on the "AI turn" for digital health: the Internet of Things and cyber-physical systems," *Omics: A Journal of Integrative Biology*, vol. 23, no. 6, pp. 308–311, 2019.

[109] X. Zhou, W. Liang, S. Shimizu, J. Ma, and Q. Jin, "Siamese neural network based few-shot learning for anomaly detection in industrial cyber-physical systems," *in IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5790–5798, 2020.

[110] K. Hanbay, M. F. Talu, and Ö. F. Özgüven, "Fabric defect detection systems and methods—a systematic literature review," *Optik*, vol. 127, no. 24, pp. 11960–11973, 2016.

[111] K. Yildiz, A. Buldu, M. Demetgul et al., "A novel thermal-based fabric defect detection technique," *Journal of the Textile Institute*, vol. 106, no. 3, 2015.

[112] B. Fang, X. Long, Y. Zhang et al., "Fabric defect detection using vision-based tactile sensor," 2020, https://arxiv.org/abs/2003.00839.

[113] S. Bangare, N. Dhawas, V. Taware et al., "Fabric fault detection using image processing," *IJARCCE*, vol. 6, pp. 405–409, 2017.

[114] X. Zhou, W. Liang, K. I.-K. Wang et al., "Academic influence aware and multidimensional network analysis for research collaboration navigation based on scholarly big data," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 1, pp. 246–257, 2021.

[115] Z. Wang and J. Jing, "Pixel-wise fabric defect detection by CNNs without labeled training data," *IEEE Access*, vol. 8, pp. 161317–161325, 2020.

[116] L. Chen, N. Yan, H. Yang et al., "A data augmentation method for deep learning based on multi-degree of freedom (DOF) automatic image acquisition," *Applied Sciences*, vol. 10, no. 21, p. 7755, 2020.

[117] W. Wei, D. Deng, L. Zeng et al., "Real-time implementation of fabric defect detection based on variational automatic encoder with structure similarity," *Journal of Real-Time Image Processing*, pp. 1–17, 2020.

[118] D. Siegmund, T. Samartzidis, B. Fu et al., "Fiber defect detection of inhomogeneous voluminous textiles," in *Proceedings of the MCPR 2017*, pp. 278–287, Huatulco, Mexico, June 2017.

[119] T. Shunji, N. Kazuki, U. Hideyuki et al., "Research into development of the defect detection system for knitted fabric produced by the circular knitting machines by image

analysis," *Journal of Textile Engineering*, vol. 64, no. 2, pp. 45–49, 2018.

[120] R. Schmitt, T. Fürtjes, B. Abbas et al., "Real-time machine-vision-system for an automated quality monitoring in mass production of multiaxial non-crimp fabrics," *IFAC-PapersOnLine*, vol. 10267, no. 3, pp. 769–782.

[121] N.-T. Huynh, "Online defect prognostic model for textile manufacturing," *Resources, Conservation and Recycling*, vol. 161, p. 104910, 2020.

[122] I. Ilhan, "Concept of industry 4.0 in textile manufacturing processes," *Pamukkale University Journal of Engineering Sciences*, vol. 25, pp. 810–823, 2019.

[123] K. Yan, W. Shen, Q. Jin et al., "Emerging privacy issues and solutions in cyber-enabled sharing services: from multiple perspectives," *IEEE Access*, vol. 7, pp. 26031–26059, 2019.

[124] X. Zhou, Y. Hu, W. Liang et al., "Variational LSTM enhanced anomaly detection for industrial big data," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3469–3477, 2020.

[125] X. Zhou, X. Xu, W. Liang et al., "Intelligent small object detection based on digital twinning for smart manufacturing in industrial CPS," *IEEE Transactions on Industrial Informatics*, p. 1, 2021.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 5518909, 14 pages
https://doi.org/10.1155/2021/5518909

Research Article

An Approach Based on the Improved SVM Algorithm for Identifying Malware in Network Traffic

Bo Liu, Jinfu Chen, Songling Qin, Zufa Zhang, Yisong Liu, Lingling Zhao, and Jingyi Chen

School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, China

Correspondence should be addressed to Jinfu Chen; jinfuchen@ujs.edu.cn

Received 3 February 2021; Revised 28 March 2021; Accepted 20 April 2021; Published 30 April 2021

Academic Editor: Entao Luo

Due to the growth and popularity of the internet, cyber security remains, and will continue, to be an important issue. There are many network traffic classification methods or malware identification approaches that have been proposed to solve this problem. However, the existing methods are not well suited to help security experts effectively solve this challenge due to their low accuracy and high false positive rate. To this end, we employ a machine learning-based classification approach to identify malware. The approach extracts features from network traffic and reduces the dimensionality of the features, which can effectively improve the accuracy of identification. Furthermore, we propose an improved SVM algorithm for classifying the network traffic dubbed Optimized Facile Support Vector Machine (OFSVM). The OFSVM algorithm solves the problem that the original SVM algorithm is not satisfactory for classification from two aspects, i.e., parameter optimization and kernel function selection. Therefore, in this paper, we present an approach for identifying malware in network traffic, called Network Traffic Malware Identification (NTMI). To evaluate the effectiveness of the NTMI approach proposed in this paper, we collect four real network traffic datasets and use a publicly available dataset CAIDA for our experiments. Evaluation results suggest that the NTMI approach can lead to higher accuracy while achieving a lower false positive rate compared with other identification methods. On average, the NTMI approach achieves an accuracy of 92.5% and a false positive rate of 5.527%.

1. Introduction

With the growth of the internet, network attacks are becoming increasingly frequent, and cyber security has become a problem that security experts urgently need to solve. Since we cannot prevent the generation of network attacks, an alternative approach is to automatically identify malware in network traffic. There are many network traffic classification methods for identifying malware in network traffic. However, these methods have two major drawbacks, i.e., incurring low accuracy and leading to high false positive rate. For this purpose, we propose an identification approach for malware in network traffic.

For machine learning-based classifiers, the first step is to extract the features of the data. However, not every feature will have the same impact on the classification. This means that some features are easier to use for classification, while others play a minimal role. Additionally, for large-scale data, we extract a very large number of features, which is not conducive to classification. Therefore, we need to preprocess the data. In this paper, we apply the stratified sampling technique to sample some data from the original dataset as experimental samples. The samples extracted by this technique will restore the features of the dataset to the maximum extent and will not generate too many redundant features. We first apply ReliefF algorithm [1] to extract the features of the network traffic in this paper. To further improve the accuracy of the classifier, we perform a dimensionality reduction operation on the extracted features. This is equivalent to turning a complex problem with high dimensionality into a simple problem with low dimensionality, which helps in classification.

Each machine learning-based classifier has its own suitable application scenario. In [2], Shafiq et al. verified that the decision tree gives the best classification results for network traffic. However, Cao et al. [3] improved the SVM algorithm to obtain higher accuracy than the decision tree. In this paper, we improve the SVM algorithm for classifying network traffic in terms of both parameter optimization and kernel function selection. In our independent experiments, on average, the radial basis function kernel can achieve an accuracy of 88.3%. Therefore, we finally choose radial basis function kernel as the kernel function of the SVM algorithm. This leads to the design and implementation of an improved SVM algorithm, called OFSVM. The accuracy of the sigmoid kernel is comparable to that of the radial basis function kernel. However, the overall result is not as good as the radial basis function kernel, i.e., with the average results, 85.92% for the sigmoid kernel vs. 88.3% for the radial basis function kernel.

This paper proposes an approach for identifying malware in network traffic by preprocessing the original dataset and applying feature extraction as well as feature dimensionality reduction to extract the major features and finally classify them using the OFSVM algorithm proposed in this paper to identify malware in network traffic. To show the usefulness of the NTMI approach, we apply it to four network traffic datasets we collected and a public common dataset. Furthermore, we compared the approach with existing identification methods (namely, SVM [4], LA-SVM [5], naive Bayes [6], and decision tree [7]). Experimental results show that the NTMI approach can achieve higher accuracy and lower false positive rate. Considering four datasets, on average, the NTMI can achieve 92.5% accuracy and 5.527% FPR.

In this paper, we present a malware identification approach and make three contributions.

First, we propose an improved SVM algorithm for more accurate classification of network traffic from two aspects: parameter optimization and kernel function selection. Additionally, we compare it with SVM algorithms using other kernel functions. Considering all together, it can be concluded that the algorithm proposed in this paper achieves the highest accuracy.

Second, we sample the original dataset and process the data using feature dimensionality reduction methods. The purpose of all these operations is to extract the major features of the data to reduce the impact of redundant and minor features on the classification results. The classification is then performed using the OFSVM algorithm proposed in this paper. Evaluation results demonstrate that the NTMI approach can lead to the highest identification performance compared to other methods.

Third, to show the usefulness of the NTMI approach, we capture real network traffic at different times of the week and selected 10% of the data in the public dataset CAIDA as the experimental data.

The rest of the paper is organized as follows. In Section 2, we present some previous methods related to network traffic classification, and in Section 3, we describe the design and implementation of the NTMI approach proposed in this paper. Section 4 introduces the experimental setup and discusses the experimental results. The conclusions are summarized in Section 5.

2. Related Work

Many scholars have already conducted research in network traffic classification or identification of malware in network traffic. Each method has its advantages and is worthy of later scholars. In this section, we present some related preliminary works.

Shafiq et al. [2] discussed network traffic classification techniques and captured real-time internet datasets. Additionally, they applied feature extraction tools to extract features and classified network traffic using four machine learning methods: support vector machine, C4.5 decision tree, naive Bayes, and Bayes net. Experimental results suggest that C4.5 decision tree can obtain a more accurate classification result compared to other classifiers. In [8], Yang et al. found that the parameters transmitted by the application layer vary according to protocols and proposed utilizing decision trees based on the minimum partition distance to perform classification. Experiments show that intercepting the first 4 or 6 packets can shorten the classification time and have higher accuracy, thus proving the effectiveness of the method. Soysal and Schmidt [9] utilized three supervised machine learning algorithms, namely, Bayes networks, decision trees, and multilayer perceptrons, for flow-based classification of network traffic. Additionally, they investigated the effect of the amount and composition of training data on the traffic classification performance. Experiments show that ML algorithms such as Bayes networks and decision trees are suitable for high-speed internet traffic classification and emphasize the importance of correctly classifying training examples.

Liu et al. [10] employed the K-means clustering algorithm to build classifiers using statistical information as input vectors. Experimental results on different datasets suggest that the method can obtain an overall accuracy of up to 80%, which increases to more than 90% after log transformation. It is demonstrated experimentally that the K-means performs well for traffic classification. Shrivastav and Tiwari [11] proposed a semisupervised method for classifying network traffic, which can design classifiers from training data consisting of only a few labeled traffic and many unlabeled flows. The method uses the K-means clustering algorithm to partition the training dataset into disjoint clusters and perform classification. Experiments demonstrate that the test error rate depends on the number of clusters randomly used in the training phase. Furthermore, the accuracy of the classifier ranges from 70% to 96% for various datasets.

Teufl et al. [12] proposed a framework to simplify empirical model selection and feature extraction, called Intelligent Feature-based Classification Tool (InFeCT). InFeCT analyzes network traffic to check whether the data in the traffic violate a certain rule and extracts the best set of features from the data to build a traffic classification model to classify the network traffic. Bekerman et al. [13] proposed an end-to-end surveillance-based system to detect unknown malware using network traffic classification. The classification method extracts behavioral features and applies feature selection methods to identify the most meaningful features while reducing the data dimensionality. The accuracy of the proposed method is experimentally demonstrated to be effective in both sandbox and real networks, and the method can detect most modern malware as well as new and unknown malware. In [14], Mu and Wu proposed a parallelized network traffic classification method based on the hidden Markov model using packet-level properties in network traffic flows. Experimental results suggest that the classification method can obtain a high accuracy, giving more than 90% accuracy on the collected dataset. The most common technique used in network traffic classification is the machine learning approach.

Sethi and Behera [15] proposed using Deep Packet Inspection algorithm for internet traffic classification. The method achieves the classification of network traffic by analyzing and processing the data based on parameters such as the data to be searched, the time of searching, available bandwidth, the number of accessing users, and architecture of the network system, using clustering methods in machine learning and signature techniques. Rezaei and Liu [16] proposed a general deep learning-based framework for traffic classification and introduced common deep learning methods as well as their application in traffic classification tasks. Lim et al. [17] proposed using the convolutional neural network and residual network for network traffic classification. Experimental results show that the use of deep learning models for network traffic classification is effective and that the residual network outperforms the convolutional neural network for classification.

3. Research Methodology

The objective of this paper is to identify malware in network traffic. For this purpose, we present a malware identification approach dubbed Network Traffic Malware Identification (NTMI). The approach consists of three steps. NTMI first extracts features from network traffic, then reduces the dimensionality of the extracted features, and finally utilizes the improved SVM algorithm for classification to identify malware in network traffic.

3.1. Feature Extraction.

The first step in classifying network traffic using machine learning techniques is to extract the features of the network traffic. In this section, we first describe how to collect, sample, and normalize the data.

3.1.1. Data Collection.

To identify malware in network traffic, we first extract features from the collected network traffic data. We use the NetFlow tool [18] to collect the network traffic data. It is a lightweight tool that monitors all traffic passing through a port during a specified time period and then gets the packet version, number, buffer size, and other information.

3.1.2. Data Sampling.

Furthermore, instead of extracting features directly from the collected network traffic, we first perform data sampling to select a better subset. The aim of data sampling is to select some data as a subset of the entire dataset and sample it for observation because the subset inherits the features of the original dataset, thus allowing the evaluation of the whole dataset. Data sampling is divided into three categories: systematic sampling, random sampling, and stratified sampling. Systematic sampling is the sampling of a portion of the total sample according to a certain sampling distance. Random sampling refers to the random selection of some sample data from the entire sample data. Stratified sampling means that the entire data sample set is first stratified according to specified rules, and then some data are randomly selected from each stratum according to a specified proportion. In this paper, we take stratified sampling to select the sample data.

3.1.3. Data Normalization.

If the selected dataset is anomalous, it will eventually affect the effectiveness of malware identification. Therefore, we need to normalize the dataset. By specifying the feature attributes of all data in a range, the normalized data can reduce the training time and improve the classification performance. If they are out of the specified interval range, the data will be excluded, thus helping to classify the network traffic, and the identification model built on this basis will improve the efficiency of identifying malware.

3.1.4. Traffic Feature Extraction.

To achieve the identification of malware in network traffic, it is indispensable to extract the feature attributes of the data transmitted in the network and build a malware identification model. Through studying the behavior of network attacks, we find that malware has some common features which help us to identify them. An attacker will attack multiple commonly used ports or ports that have been closed in a short period of time, for example, anomalous packets that send only SYN or FIN packets, a great number of false connections or REJ packets, and plenty of network traffic packets. If common features of network traffic data packets can be extracted, it will improve the accuracy of malware identification.

Commonly used feature extraction methods are SNMP protocol technology [19] and probe technology [20]. SNMP protocol technology monitors the network links, but the features obtained by this technology are too few to be classified. Probe technology can be applied to the links of network traffic to obtain the traffic features quickly. However, it is not suitable for large-scale traffic feature extraction and is too time consuming. Additionally, the technique focuses on the extraction of protocol features, which cannot accurately parse the packet messages' information. This paper applies the ReliefF algorithm [1] for feature extraction, which is superior to the aforementioned common feature extraction methods. This technology compares the correlation of sample types and feature attributes on the processed dataset. The weight will keep increasing as the correlation becomes higher, and a threshold is set. If the weight corresponding to the correlation between the feature attribute and the sample type exceeds the set threshold, we keep the feature attribute; otherwise, we discard the attribute.

Furthermore, if more than one feature attribute appears in a packet, the feature attribute that appears most frequently is selected. The specific feature selection process is as follows:

(i) Select some samples s randomly from the dataset D in a stratified sampling manner

(ii) Select y samples r of the same type D_a nearest to the sample s

(iii) Select y samples t from different types of D_b

(iv) Calculate the distance between sample s and sample r as D_{sr} and the distance between sample s and sample t as D_{st}

If $D_{sr} > D_{st}$, it means that the feature attribute is problematic and cannot be utilized for classification, and we set a smaller weight value; otherwise, this feature attribute is helpful for classification, and we set a larger weight value. The formula for calculating the feature weights is shown in equation (1), where $w(x)$ is the corresponding weight, $\mathrm{d}(x, r, t)$ represents the Euclidean distance of sample r, sample t, and feature x, D_j is the j-th sample data in the dataset, and n refers to the calculation of the weight size in n data for feature extraction. Loop through the above process, and the final computed weights are compared with the set threshold. If the requirements are met, the feature attribute is retained; otherwise, it is discarded. Finally, we can get the set S of extracted feature attributes.

$$w(x) = \sum_{i=1}^{n} \frac{\mathrm{d}(x, r, t)}{yn} + \sum_{D_j \in D} \frac{\left(1/\left(1 - p\left(D_j\right)\right)\right) \sum_{i=1}^{n} \mathrm{d}(x, r, t)}{yn}. \quad (1)$$

Through the ReliefF algorithm [1], Table 1 records some of the extracted network traffic feature attributes.

3.2. Feature Dimensionality Reduction. After feature extraction is performed on network traffic, a certain traffic data packet contains a variety of feature attributes, which poses a complex high-dimensional feature space problem for the classification of network traffic. Some redundant features not only lead to increased learning complexity of the classification algorithm but also cause overfitting and local optimization problems. When the proportion of key features for malware identification is small, the final identification result will be poor. To solve the above problems, this study chooses key feature combinations to achieve dimensionality reduction of traffic features to help build the corresponding malware identification model.

The extracted feature attributes are first added to the set S. We propose utilizing the filter feature dimensionality reduction method with the help of the information gain technology [21], i.e., $E_{\mathrm{IG}} = \mathrm{evaluate}(F_{\mathrm{filter}}, S)$. The algorithm is an evaluation of the information gain on the set S of feature attributes. By evaluating the impact of each feature attribute on the subsequent classification, it is determined whether to update the value of E_{IG} and the feature attribute set S, where the information gain value $\mathrm{Test}_{\mathrm{IG}}$ of the candidate feature subset is calculated. When $\mathrm{Test}_{\mathrm{IG}} > \mathrm{Test}_{\mathrm{best}}$, the evaluation value and feature subset will be updated; otherwise, they will not be updated. And then, the heuristic search strategy [22] is used to sort the feature attributes to obtain the feature attribute set S_1. The process is repeated until the specified number of times is reached. Based on this, we employ the wrapper method [23] for secondary feature selection, and the heuristic sequence forward search method is used to obtain the feature attribute set S_2. After the feature dimensionality reduction, it not only reduces the time and computational complexity but also improves the classification effect.

When using the wrapper method, equation (2) calculates the correlation of the traffic feature attributes to perform a secondary selection of the feature attributes, where n represents the number of feature attributes for all initial selections, k_∂ represents the feature attribute coefficient, m_{ri} represents the average of the traffic feature attribute for the i-th packet, $h_{r\,\partial}^2$ is the corresponding variance, and m_r represents the average of the traffic feature attribute r.

$$F_r = \frac{1}{k_\partial} \cdot \frac{\sum_{i=1}^{n} k_\partial \left(m_{ri} - m_r\right)^2}{\sum_{i=1}^{n} k_\partial h_{r\partial}^2}. \quad (2)$$

After performing feature dimensionality reduction by using the above method, the previously extracted feature set can be further simplified to eliminate redundant features. Additionally, some of the features selected in this way are almost uncorrelated with each other, which is more conducive to classification. The final feature dimensionality reduction set is presented in Table 2.

We obtain a subset of feature attributes after feature dimensionality reduction. However, these feature attributes with different units and measurement criteria are not related to each other. Therefore, this study proposes to normalize the subset of feature attributes.

The specific normalization process is as follows. We utilize min-max normalization to process the data. Linear transformation on the acquired feature subset is performed to convert the target dataset into between 0 and 1 [11], using the conversion function as follows:

$$f' = \frac{f - \min}{\max - \min}. \quad (3)$$

In this formula, min refers to the minimum value of the sample data, and max refers to the maximum value of the sample data. However, this method has the disadvantage that continuing to add data to it during the target transformation will cause max and min to be changed, thus affecting the normalization criteria. Therefore, before the normalization process, it is necessary to ensure that the dataset will remain constant.

3.3. OFSVM Algorithm. This research will improve the existing support vector machine (SVM) classification method [23] and finally realize the improvement of the classification of the program in the network traffic. The first section introduces the shortcomings of the current SVM algorithm for classifying network traffic, and the second section proposes the improvement of the algorithm.

Table 1: Extracted feature attributes.

Feature name	Feature description
origin_ip	Source IP address
destination_ip	Destination IP address
port_number	Port number
duration	Connection duration
protocol_type	Protocol type
service	Type of network service of the destination host
flag	Connection normal or error state, and this field is discrete type
src_bytes	Number of bytes of data from the source host to the destination host
dst_bytes	Number of bytes of data from the destination host to the source host
wrong_fragment	Number of wrong fragments, and this field is continuous type
urgent	Number of urgent packages, and this field is continuous type
dst_host_srv_error_rate	Percentage of connections with SYN errors
hot	Number of accesses to sensitive files and directories on the system
mark_status	Mark status
packet_rate	Packet sending rate
max_pktLens	Maximum message length
min_pktLens	Minimum message length
num_compromised	Number of occurrences of compromised condition
num_access_files	Number of access control files
same_srv_rate	Percentage of connections with the same service as the current connection
dst_host_srv_count	Number of connections with the same destination host service as the current connection

Table 2: Features after dimensionality reduction.

Feature name	Feature description
origin_ip	Source IP address
destination_ip	Destination IP address
duration	Connection duration
flag	Connection normal or error state, and this field is discrete type
src_bytes	Number of bytes of data from the source host to the destination host
dst_bytes	Number of bytes of data from the destination host to the source host
wrong_fragment	Number of wrong fragments, and this field is continuous type
mark_status	Mark status
packet_rate	Packet sending rate
max_pktLens	Maximum message length
min_pktLens	Minimum message length
same_srv_rate	Percentage of connections with the same service as the current connection
dst_host_srv_count	Number of connections with the same destination host service as the current connection

3.3.1. Existing SVM Algorithm and Its Shortcomings. SVM is a model for binary classification which is mainly used to find the maximum interval in the feature space. The objective of the SVM is to find a hyperplane in all sample data so that the distance between the nearest data on both sides and the plane is the largest. The SVM algorithm can divide the data in the training set by separated hyperplanes, of which there may be an infinite number, but the one that makes the maximum interval is selected.

In network traffic, we assume that the current network traffic set is $N = \{N_1, N_2, \ldots, N_k\}$, and its corresponding feature set is $F = \{F_1, F_2, \ldots, F_k\}$. Then, the SVM algorithm is used to construct the network traffic classification model and implement the classification of network traffic, i.e., malware or nonmalware. The SVM classification method can choose a relatively optimal classification plane to build a model for the classification and can complete a relatively stable classification under the condition of unknown sample classification. There is a lot of noise in the real network environment and a lot of unprocessed redundant features in the sample data, both of which lead to low accuracy of the classification results. In this paper, we propose to optimize the SVM algorithm in terms of parameter optimization and suitable kernel functions. The method utilizes grid search parameter optimization [24] to prevent overfitting in order to find the optimal solution. Additionally, we introduced fuzzy factors to improve the accuracy of the classification [25]. This study uses the distance from the sample to the classification hyperplane to design the fuzzy factor because this approach removes the effect of noise while reducing the effect of classification plane shape on accuracy. And then, we use the feature validity [26] to eliminate the effect of redundant features. Finally, considering the importance of kernel function parameters on the classification performance, this paper chooses the radial basis kernel function [27] to optimize the SVM algorithm.

3.3.2. Parameter Optimization. Several researchers have already improved the classification capability of the SVM algorithm, for example, genetic algorithm [28], particle swarm algorithm [29], and artificial fish swarm algorithm [30]. However, these classification algorithms still have some deficiencies in terms of stability and accuracy. Therefore, we present a new algorithm to improve the SVM algorithm, called OFSVM algorithm. This study will fully consider the complexity of real network traffic and the reasons for the decline in identification accuracy.

Parametric optimization of the SVM is mainly to find a convergent optimal solution in a finite number of searches using some search strategy in a space of many parameters. In this step, we consider two important parameters: the kernel function parameter and the penalty parameter. Among them, the penalty parameter will play a decisive role in the generalization capability of the SVM hyperplane, which is mainly used to indicate the fault tolerance when constructing the hyperplane. And the kernel function parameter will determine the scope of action, which will also affect the generalization capability of the SVM. Therefore, with the aim of finding the optimal parameter combination in a limited number of searches, we propose to employ grid search to optimize parameters to improve the SVM algorithm.

The principle of grid search used in this paper is as follows, and it consists of four main steps, which we have briefly summarized:

(i) Delineating the k-dimensional parameter space, where grid nodes are used to represent the candidate parameters
(ii) Sampling at the specified step and generating the corresponding set $P(c_i) = \{P(c_1) \times P(c_2) \times \cdots \times P(c_k)\}$
(iii) Setting the range of the parameter c_i to generate grids with different orientations
(iv) Evaluating each grid node c_i according to the specified evaluation method, and outputting the final approximate optimal solution

In this process, the incremental is first set to be times the default step size q, i.e., $q \cdot t$. This step is to reduce the search time and the density of the generated grid. Then, all sample data are searched iteratively to obtain the optimal combination of parameters. To express the fault tolerance of the sample data when constructing the classification plane, a penalty parameter P is introduced and compared with the set overfitting threshold f. When P is less than f, narrow the search space and set the step size of the search to half of the initial step size, and search again. The reduction of step length is to expand the density of the grid and thus achieve a more accurate search. If P exceeds f, expand the search space and adjust the search direction for another search. The purpose of this step is to optimize the parameters and prevent overfitting. Looping through the sample data until the penalty parameter P is within the critical range, the optimal parameter combination value is outputted. The algorithm has a large searchable space, and the nodes are uncorrelated with each other, so it is more generalizable.

To further improve the classification accuracy, fuzzy factors are first introduced. In this study, the distance from the sample to the classification hyperplane will be used to design the fuzzy factor, which will reduce the impact of the classification plane shape on the classification accuracy. On this basis, the corresponding classification hyperplane is constructed firstly. And then, the distance from each sample node to the hyperplane is calculated so that the fuzzy factor can be used to eliminate the effect of excess noise. Accordingly, it is proposed to construct feature validity to eliminate the influence of redundant features on classification accuracy.

For each sample point s_i, there is a corresponding fuzzy factor e_i, which represents the uncertainty of the sample distribution, where $0 \leq e_i \leq 1$. R^+ and R^- represent the mean point of the positive and negative samples, and normal vector can be expressed by $\alpha = \sqrt{R^+ - R^-}$. According to the method in [31], the corresponding hyperplane can be expressed as $(s - R)^2 \cos \alpha^T = 0$. In this way, the distance from the sample point to the hyperplane is described in equation (4), and then the maximum distance d_1 from the positive sample point to the hyperplane can be obtained if and only if R is R^+. In the same way, when R is R^-, d_2 is the maximum distance from the negative sample point to the hyperplane. Then, the regulatory factor ∂ is used to make $0 < e_i \leq 1$. The fuzzy factor is shown in equation (5), where the value of d is d_1 and d_2, respectively, in different positive and negative samples. Thus, the effect of excess noise on the classification accuracy is eliminated by using different fuzzy factors. Because the impact of different features on the classification is not considered, this paper proposes to introduce feature validity to eliminate the impact of weakly correlated features on classification accuracy.

$$d = \frac{(s_i - R)^2 \cos \alpha^T}{\|\alpha\|^2}, \tag{4}$$

$$e_i = \frac{\cos \alpha^T}{d\, \partial} - \ln \partial. \tag{5}$$

In [26], for each feature i of the sample data, it has a corresponding feature validity h_s^i, which can indicate the degree of influence of a certain feature used for classification. The greater the classification capability of feature i, the greater its feature validity h_s^i. In feature set S, the classification effect of each feature is judged by calculating the enhanced learning capability of each feature. If the training sample set S has a total number of $|S|$ and there are p feature attributes in a sample, the feature validity can be expressed as equation (6). When the enhancement learning value of a certain feature i is relatively large, its feature validity is relatively large, that is, its contribution to the classification will be relatively high. Finally, considering the importance of kernel function parameters on the classification performance, this study optimizes the SVM algorithm by selecting the appropriate kernel functions.

$$h_s^i = \frac{|S|^2 \ln p}{\sum_{i=1}^{n} |S| - p^2}. \qquad (6)$$

3.3.3. Appropriate Kernel Functions. The kernel function is mainly used to map the original nonlinear sample data into the feature space and then convert the nonlinear sample into a linear classifiable problem by means of the constructed optimal classification plane, thus avoiding the huge amount of calculation of the high-dimensional feature space. Assume the input space is $P \in R^n$, and the corresponding feature space is F. When there is a mapping function $\gamma(y) = Y \longrightarrow P$ and any y_i and y_j belonging to Y satisfy $K(y_i, y_j) = \gamma(y_i)^T \gamma(y_j)$, there is a kernel function K. The kernel function needs to satisfy Mercer's theorem [32], that is, for any vector in the input space, the corresponding kernel matrix should be a positive semidefinite matrix. After selecting the appropriate kernel function, the linear classification can be completed without increasing the complexity. Therefore, the classification effect of the SVM is greatly related to the kernel function. In this paper, we choose the radial basis kernel function as the kernel function. This function has good performance in the local range, and it can achieve a high classification efficiency for the sample points in the dataset. Furthermore, it is not constrained by the number of samples and feature dimensions. And the radial basis kernel function has fewer parameters, which makes the kernel function have a lower complexity. Algorithm 1 describes the improved algorithm OFSVM.

By improving the SVM algorithm in the above manner, the error is relatively small, and the identification of malware in network traffic is further improved. The input of the algorithm is the set of feature attributes to be trained as support vectors. The algorithm applies grid search for correlation search, which expands the search space and search density, and then completes the accurate search. The distance between each sample and the class is used as a fuzzy factor, and the proposed feature validity is used to eliminate the effect of redundant features on the classification accuracy. It also depends on the radial basis kernel function verified by experiments and elaborated in Section 4.3. The kernel function has a higher accuracy and is more stable and finally generates a classifier model. Its time complexity is $O(n * m)$, where n is the number of input sample feature attributes and m is the number of kernel function operations.

3.4. NTMI Approach. In the previous sections, we introduced how to extract the features of network traffic and reduce the dimensionality of the extracted features, respectively, and then overviewed the solutions for identifying malware in network traffic. Additionally, to address the inaccuracy of the traditional SVM classification, we present the OFSVM algorithm for classifying network traffic in terms of parameter optimization and appropriate kernel functions. In this section, we detail the identification model building process and propose the identification approach for malware in network traffic, i.e., NTMI.

To identify malware, the first step is to solve the problem of accurate classification in network traffic. We first apply the NetFlow tool [18] to collect real network traffic. Second, the collected network traffic data are sampled and normalized to obtain a more valuable dataset for the experiment, while the processed data are more convenient for feature extraction. Third, we utilize the ReliefF algorithm [1] to extract the features of the data packets in the network traffic. Meanwhile, the extracted features still contain some redundant feature attributes. These feature attributes will greatly reduce the accuracy of network traffic classification. Therefore, we propose to reduce the dimensionality of the above extracted feature set. Feature dimensionality reduction consists of a total of 4 steps:

(i) Calculating and evaluating each feature using the information gain technology
(ii) Sorting the feature set
(iii) Selecting secondary features using the wrapper method
(iv) Calculating the correlation of the features adopting the heuristic sequence forward search method

Next, the obtained feature subset needs to be normalized, and all feature attributes are converted into numerical values which are then put into a matrix array to calculate the minimum Euclidean distance. Then, the OFSVM algorithm is used to train for a better classifier with the processed network traffic testing set as the input. This classifier can classify normal programs and malware in network traffic and finally identify malware in network traffic. Algorithm 2 describes the specific NTMI approach.

The input of the algorithm is the set of collected traffic packets, and the final output is the dataset of malware in network traffic. The time complexity in the process of feature extraction and feature dimensionality reduction is greater than the time for feature normalization, so the final time complexity of the algorithm is $O(p * k)$, where p is the number of data packets normalized and k is the number of extracted feature attributes, which can be approximated as $O(n^2)$. The NTMI approach is less costly compared to several other classification methods in the experimental section.

4. Experiments and Discussion

To verify the effectiveness of the NTMI approach for identifying malware in network traffic, we compare it with existing identification methods, i.e., SVM [4], LA-SVM [5], naive Bayes model (NBM) [6], and decision tree model (DTM) [7]. We conduct experiments on each of the five datasets selected for this paper. To avoid errors caused by a single experiment, we perform 100 experiments for each method separately and calculated the average accuracy and average false positive rate of the method as the final experimental results.

```
     Input: executedDataM//the set of processed feature attributes
     Output: generatedClassifier//the generated optimized classifier
(1)  Construct fuzzyFactor = null;//calculate the distance between each sample and the class as a fuzzy factor to improve the
     classification accuracy
(2)  Construct executedDefaultStep = q, executedSearchStep = null;//control search time and grid density
(3)  Construct executedPenaltyParameter;//express the fault tolerance of the sample data when constructing the classification plane of
     SVM
(4)  Construct executedOverfittingThreshold = f;//judge whether the penalty parameter is within the critical range
(5)  representCandidateParameters();//use grid nodes to represent candidate parameters
(6)  set the range of parameters to generate grids in different directions;
(7)  for each sample i in executedDataM do
(8)    Construct executedSearchStep = q.t;// the incremental step is t times the default step q
(9)    constructTraverseSearch();//perform traversal search on all samples
(10)   divide into i-dimensional parameter space among i parameters;
(11)   if (executedPenaltyParameter(i) < executedOverfittingThreshold) then
(12)     executedSearchStep = 2/q;// reduce the step size to increase the grid density for a more accurate search
(13)     constructTraverseSearch();//perform traversal search on all samples
(14)   else
(15)   expand the search space and adjust the search direction;
(16)   constructTraverseSearch();//perform traversal search on all samples
(17)   end if
(18)   panel = createClassificationHyperplane();// construct the corresponding classification hyperplane
(19)   calculateDistance(M[i], panel);// calculate the distance between each sample node and the hyperplane as a fuzzy factor
(20)   computeFeatureValidity(i);// calculate the feature i of each sample data, which has a feature validity, and determine the
       classification effect of each feature
(21)   useRadialBasisKernel();// the kernel function has lower complexity and higher classification efficiency
(22) end for
(23) generateClassification();// generate the optimized classifier
(24) return generatedClassifier;
```

Algorithm 1: OFSVM algorithm.

```
     Input: executedOriginalData// the set of collected traffic data packets
     Output: identifyMaliciousData// the set of identified malware
(1)  Construct executedOriginalFeatureSet = nulll// store feature attributes extracted from network traffic packets
(2)  Construct identifyMaliciousData = null;// the set of identified malware
(3)  Construct executedNormalizationData = null;// store normalized data
(4)  executedOriginalData = collectNetworkFlow();// use NetFlow to collect data packets for assignment
(5)  for each data package p in $executedOriginalData_{train}$ do
(6)    executedNormalizationData = dataNormalization();// to complete data sampling and normalization
(7)  end for
(8)  for each data package p in executedNormalizationData do
(9)    executedOriginalFeatureSet = useReliefFCompleteFeatureExtracted ($executedNormalizationData_p$);
(10)   for each feature kexecutedOriginalFeatureSet do
(11)     temp = compare($executedOriginalFeatureSet_k$, $\partial$);// compare each extracted feature attribute k with a threshold $\partial$ and return
         the value temp
(12)     if (temp = = 1) then
(13)       deleteFeature($executedOriginalFeatureSet_k$);// delete this feature attribute
(14)     end if
(15)   end for
(16)   executedFirstFeatureSet = outputFeatureExtraction();// retain the feature attributes extracted from each packet
(17) end for
(18) for each feature j in executedFirstFeatureSet do
(19) use information gain technology to calculate and evaluate each feature;
(20) normalizedFeature = sencondExtraction($executedFirstFeatureSet_j$); // sort feature attributes and use Wrapper for second feature
     extraction
(21) end for
(22) realizeUnit();// convert to unitless values and keep the data at the same order of magnitude
(23) classifyModel = useOFSVMAlgorim(normalizedFeature);// generate the classification model
(24) identifyMalware (classifyModel, $executedOriginalData_{test}$);
(25) return identifyMaliciousData;
```

Algorithm 2: NTMI approach.

4.1. Experimental Datasets. We capture four sets of network traffic data at different periods within a week, called NTDS1, NTDS2, NTDS3, and NTDS4, respectively. These four datasets have a relatively large randomness so that the performance of the proposed method can be better judged. Meanwhile, we adopt the public dataset CAIDA [33] to train and test the above methods. Due to the overwhelming amount of data in this dataset, we randomly select 10% of the data for the experiment. Table 3 summarizes the specific information of the network traffic datasets.

4.2. Experimental Metrics. We use accuracy and false positive rate (FPR) as experimental metrics for the experiment. Equations (7) and (8) show the calculation of accuracy and FPR. In equations (7) and (8), TP represents the number of samples that are correctly identified as malicious traffic. FP indicates the number of samples that are misclassified as the normal traffic, referring to abnormal traffic but mistakenly considered normal. FN denotes the number of samples that are misreported, i.e., normal traffic is misidentified as abnormal traffic. And TN means that the classification result is consistent with expectations, i.e., nonmalicious traffic is classified as normal traffic.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{7}$$

$$\text{FPR} = \frac{FP}{FP + TN}. \tag{8}$$

4.3. Experimental Result. In this paper, experiments are conducted using the above five feature-processed datasets to detect the effect of different kernel functions on the classification effectiveness of the proposed OFSVM algorithm. According to Table 4, for our collected datasets and the publicly available dataset CAIDA, the linear kernel and polynomial kernel have poorer accuracy than the sigmoid kernel and the radial basis function (RBF) kernel, which is mainly due to nonlinear features and higher-dimensional features. Furthermore, it can be seen from Table 4 that the sigmoid kernel function is relatively stable, which is only slightly worse than the RBF kernel. The kernel function has relatively high requirements on the parameters. From the average value of the classification accuracy of the five datasets, the classification effect of the linear kernel function is the worst (i.e., 65.92% for average accuracy), and the RBF kernel performs best (i.e., 88.3% for average accuracy), which is more stable and more suitable for nonlinear high-dimensional feature space with a lower complexity. Therefore, this study selects the RBF kernel as the kernel function of classification.

4.4. Experimental Discussion. Table 5 records the accuracy and FPR of the NTMI approach and other identification methods. From the perspective of accuracy, the NTMI approach is the algorithm with the highest accuracy, followed by DTM, NBM, LA-SVM, and SVM. In terms of

Table 3: Network traffic datasets.

Dataset	Training set		Testing set	
	Nonmalware	Malware	Nonmalware	Malware
NTDS1	17,296	5,690	7,435	4,709
NTDS2	15,735	3,972	8,395	3,674
NTDS3	12,947	4,468	7,083	4,238
NTDS4	19,392	4,938	8,974	3,950
CAIDA	170,874	9,763	29,047	11,933

Table 4: Accuracy of OFSVM using different kernel functions.

Dataset	Accuracy (%)			
	Polynomial	Sigmoid	Linear	RBF
NTDS1	75.4	87.9	67.4	90.3
NTDS2	86.3	89.3	55.3	88.4
NTDS3	53.8	82.4	64.2	89.3
NTDS4	91.4	87.3	77.3	87.1
CAIDA	72.3	82.7	65.4	86.4
Average	75.84	85.92	65.92	88.3

Table 5: Comparison of accuracy and FPR on four collected datasets.

Dataset	Metric (%)	SVM	LA-SVM	NBM	DTM	NTMI
NTDS1	Accuracy	82.6	84.2	83.2	87.2	92.4
	FPR	12.45	11.90	9.56	9.47	5.78
NTDS2	Accuracy	82.8	84.8	84.3	87.5	92.5
	FPR	11.34	10.58	9.73	9.36	5.74
NTDS3	Accuracy	84.3	84.4	86.2	85.9	92.0
	FPR	10.87	9.94	9.46	10.18	5.41
NTDS4	Accuracy	84.6	85.6	86.2	86.6	93.1
	FPR	11.08	8.69	10.14	9.92	5.18
Average	Accuracy	83.57	84.75	84.97	86.8	92.5
	FPR	11.435	10.277	9.722	9.732	5.527

average results, we can observe that the NTMI approach is optimized by 8.93% compared to the SVM algorithm; NTMI is optimized by 7.75% compared to the LA-SVM algorithm; NTMI approach is optimized by 7.56% compared to the NBM algorithm; and NTMI is optimized by 5.7% compared to the DTM algorithm. As can be seen from Table 5, the NTMI approach is more accurate than the other four methods in identifying malware in network traffic.

For the FPR of these five identification methods, our proposed NTMI approach can achieve the lowest false positives. On average, the FPR of the NTMI approach is only 5.527%. The lowest FPR among other methods is 9.722% for NBM. Therefore, the NTMI approach proposed in this paper is the most effective for the identification of malware in network traffic, both in terms of accuracy and FPR.

To illustrate the effectiveness of the NTMI approach more visually, we plot the accuracy and FPR curves of these five identification methods. Figures 1–4 depict the accuracy curves of these methods. As can be seen from these figures, compared with the NTMI approach, the identification performance of the other four methods decreases significantly as the number of data packets continues to increase, and it is expected that when the number of data packets

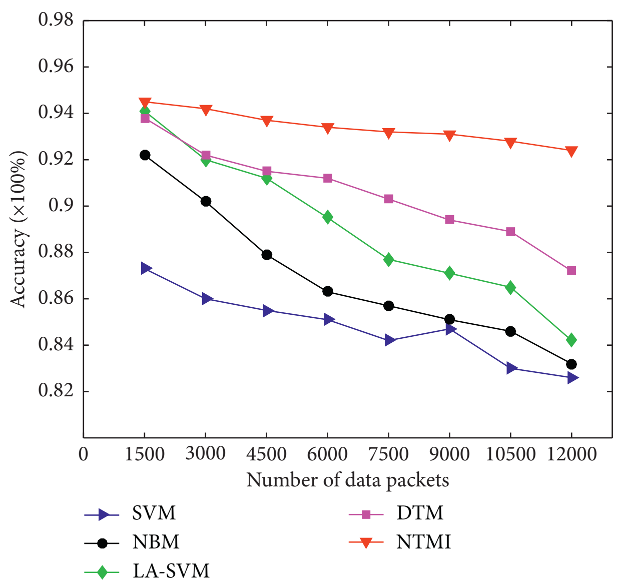


Figure 1: Comparison of accuracy on NTDS1.

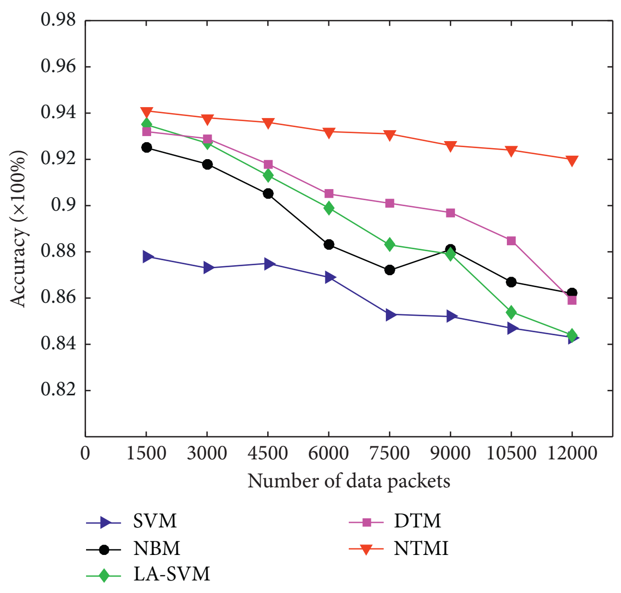


Figure 3: Comparison of accuracy on NTDS3.

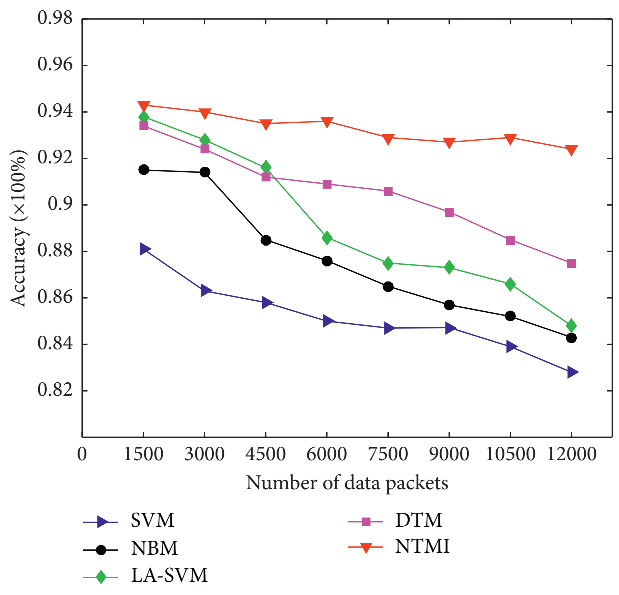


Figure 2: Comparison of accuracy on NTDS2.

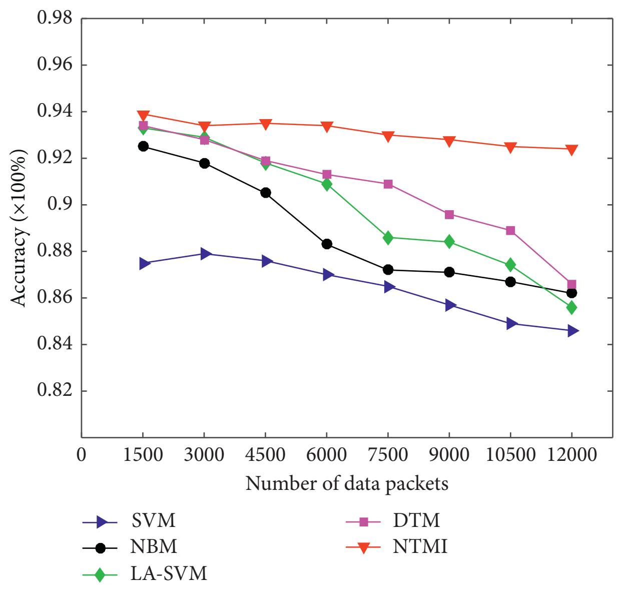


Figure 4: Comparison of accuracy on NTDS4.

expands exponentially, the accuracy of their identification will drop again.

As can be seen from Figures 5–8, compared with the other four methods, the NTMI approach has the lowest FPR in malware identification. The approach has not only high accuracy but also low FPR, which also shows that its identification effect is the best. With the increasing number of data packets, the FPR of the current five methods shows an increasing trend, but the FPR of the NTMI approach tends to be relatively stable. When the number of testing sets increases to about 12,000, it eventually stabilizes at 5.527%. Meanwhile, the NTMI approach consumes less time overhead when identifying malware. Therefore, the NTMI approach proposed in this paper has better identification effectiveness and more stable performance in both accuracy and FPR.

To better verify that the NTMI approach proposed in this paper has good universality, this study selects the widely used dataset CAIDA for experiments. Figures 9 and 10 summarize the accuracy and FPR of the above five identification methods, and the following conclusions can be drawn.

We select 10% of the dataset for training and testing because the public dataset is large, and the final testing dataset is close to about 40,000. As can be seen from Figure 9, compared with other methods, the accuracy of the NTMI

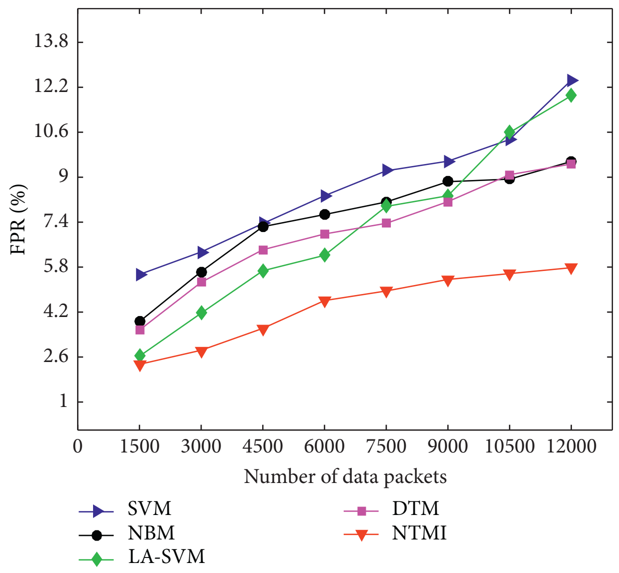


Figure 5: Comparison of FPRs on NTDS1.

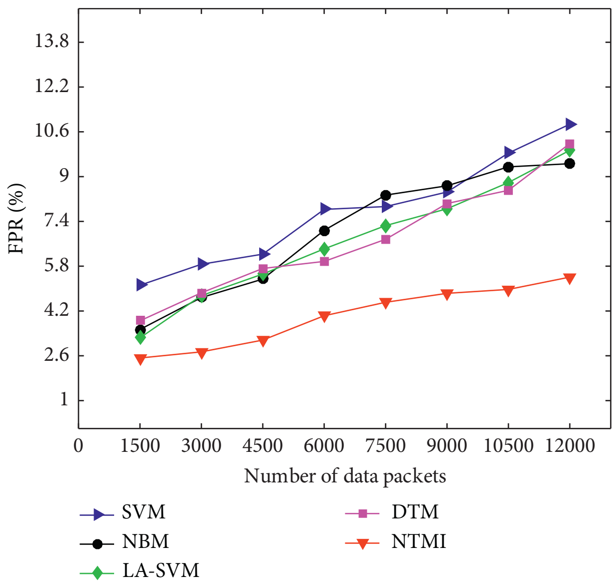


Figure 7: Comparison of FPRs on NTDS3.

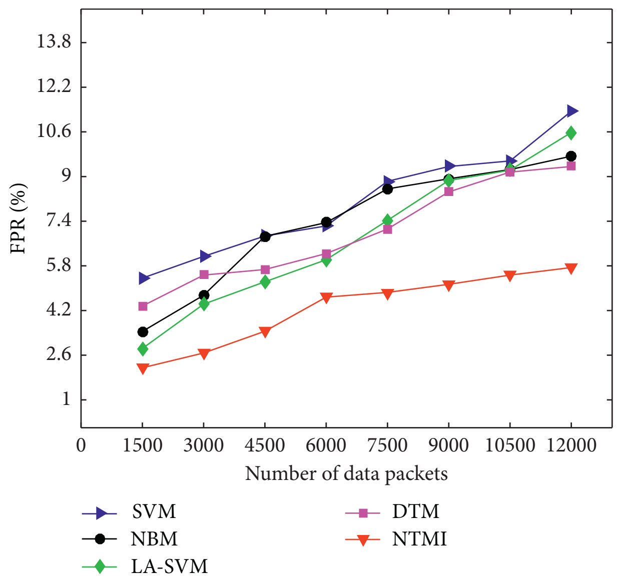


Figure 6: Comparison of FPRs on NTDS2.

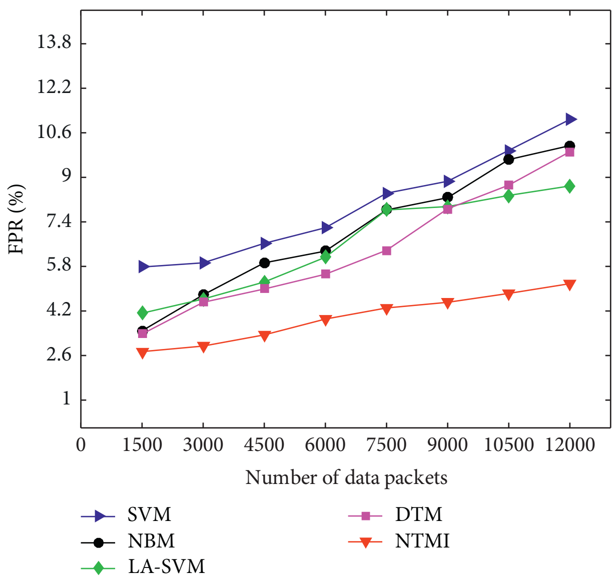


Figure 8: Comparison of FPRs on NTDS4.

approach still performs well. The accuracy of the NTMI approach can reach 91.7%, while the highest accuracy achievable by other classification methods is 78.6%. As the number of data packets continues to increase, the accuracy of the remaining identification methods has dropped significantly.

As can be seen from Figure 10, the FPR of the NTMI approach is lower than that of the other four methods for larger public datasets of network traffic and tends to be stable, remaining around 6%. Compared with the NTMI approach, the other four methods have in common that the FPR is increasing, which makes the identification performance more unreliable. Additionally, the final time overhead of the NTMI approach is between 30 s and 50 s faster than other methods. As can be seen from Table 5, for the accuracy and FPR, the results of the NTMI approach on the datasets collected in this paper are very similar to the public dataset CAIDA, which also proves the proposed approach is feasible in the real network environment. Ultimately, we conclude that the NTMI approach can achieve the highest identification performance and is practical.

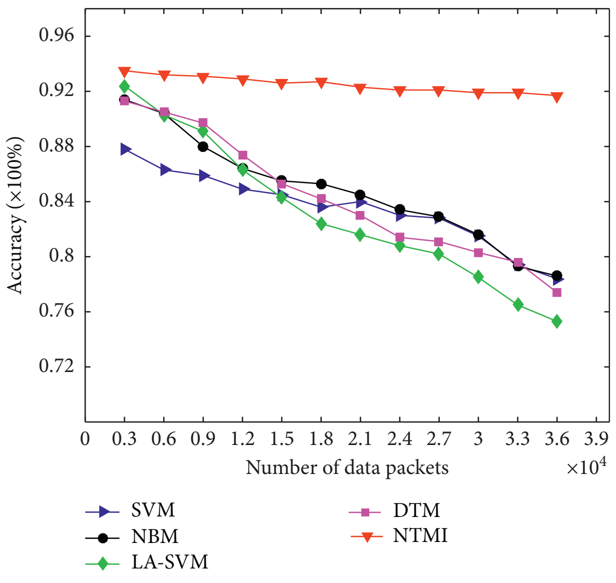


Figure 9: Comparison of accuracy on CAIDA.

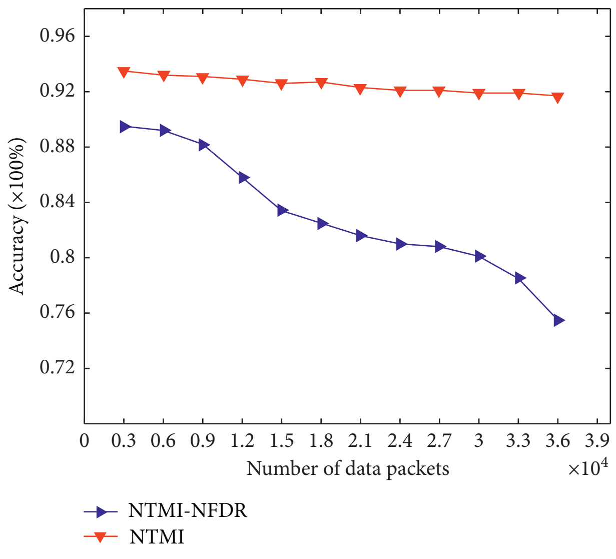


Figure 11: Accuracy of feature dimensionality reduction.

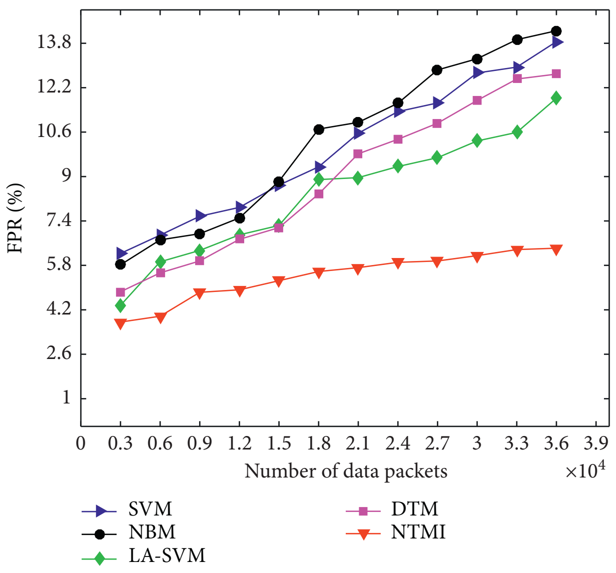


Figure 10: Comparison of FPRs on CAIDA.

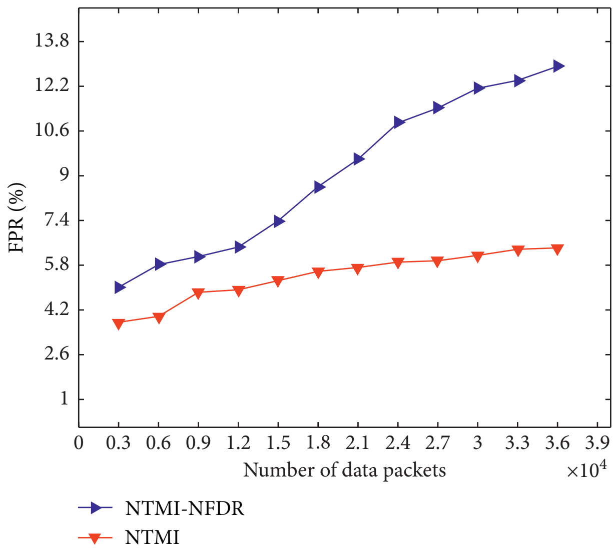


Figure 12: FPRs of feature dimensionality reduction.

4.5. Effectiveness of NTMI. To better measure the effect of the proposed method on the classification performance in this paper, we compare the accuracy and FPRs of the NTMI approach when performing classification and without feature dimension reduction, respectively. NTMI-NFDR is denoted as an identification approach that does not perform feature dimensionality reduction. We perform comparison experiments of NTMI and NTMI-NFDR on the public dataset CAIDA. Figures 11 and 12 plot the experimental results. We can observe that the NTMI approach performs better in terms of accuracy and FPR after feature dimensionality reduction on the extracted feature attribute set, which also verifies that the feature dimensionality reduction method extracted in this paper is feasible.

5. Conclusion

To identify malware in network traffic, we first sample and normalize the data. Secondly, we extract features from the processed data and reduce the dimensionality, thus eliminating the impact of some redundant features on the classification performance of network traffic. And then, we present the OFSVM algorithm based on the SVM algorithm for classifying network traffic and improving the accuracy of classification in network traffic. The OFSVM algorithm improves the SVM algorithm in terms of both parameter optimization and kernel function selection. Eventually, we propose the NTMI approach for identifying malware in network traffic. Experimental results show that the NTMI approach can achieve higher accuracy and lower FPR compared with other identification methods.

To verify the effectiveness of the NTMI approach, we compare it with four other classification methods, i.e., SVM, LA-SVM, NBM, and DTM. Additionally, we evaluate these five methods on five datasets. Evaluation results suggest that the algorithm proposed in this paper outperforms the other four classification methods in terms of both accuracy and false positive rate. Its average accuracy reaches 92.5%, while the average false positive rate is only 5.527%. In the publicly available dataset CAIDA, our proposed NTMI approach achieves the highest accuracy and the lowest false positive rate, i.e., 91.7% for accuracy and 6.42% for FPR. Therefore, the experimental results can prove the effectiveness of the algorithm.

However, the NTMI approach proposed in this paper is currently not a completely perfect classifier. Future research will consider whether it is possible to further detect which vulnerabilities are exploited by the identified malware. This could help security experts to be able to effectively identify the type of attack and provide solutions quickly.

Data Availability

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

Disclosure

A preliminary version of this paper was presented at the 7th International Conference on Dependable Systems and Their Applications (DSA 2020) (Qin et al., 2020).

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was partly supported by the National Key R&D Program of China (Grant no. 2020YFB1005500), the National Natural Science Foundation of China (NSFC) (Grant nos. U1836116 and 61872167), and the Leading-edge Technology Program of Jiangsu Natural Science Foundation (Grant no. BK20202001).

References

[1] M. Robnik-Šikonja and I. Kononenko, "Theoretical and empirical analysisof Relief F and R Relief F," *Machine Learning*, vol. 53, no. 1, pp. 23–69, 2003.

[2] M. Shafiq, X. Yu, A. A. Laghari et al., "Network Traffic Classification techniques and comparative analysis using Machine Learning algorithms," in *Proceedings of the 2nd IEEE International Conference on Computer and Communications (ICCC)*, pp. 2451–2455, Chengdu, China, October 2016.

[3] J. Cao, Z. Fang, G. Qu, H. Sun, and D. Zhang, "An accurate traffic classification model based on support vector machines," *International Journal of Network Management*, vol. 27, no. 1, 2017.

[4] Y. Zhang, B. Li, H. Lu, A. Irie, and X. Ruan, "Sample-specific SVM learning for person Re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1278–1287, Las Vegas, NV, USA, June 2016.

[5] Z. Ghaemi, A. Alimohammadi, and M. Farnaghi, "LaSVM-based big data learning system for dynamic prediction of air pollution in Tehran," *Environmental Monitoring and Assessment*, vol. 190, no. 5, pp. 1–17, 2018.

[6] I. Rish, "An empirical study of the naive Bayes classifier," in *Proceedings of the IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, vol. 3, no. 22, pp. 41–46, Seattle, WA, USA, August 2001.

[7] B Kamiński, M. Jakubczyk, and P. Szufel, "A framework for sensitivity analysis of decision trees," *Central European Journal of Operations Research*, vol. 26, no. 1, pp. 135–159, 2018.

[8] Z. Yang, L. Z. Li, Q. J. Ji, and Y. Q. Zhu, "Network traffic classification using decision tree based on minimum partition distance," *Journal of China Institute of Communications*, vol. 33, no. 3, pp. 90–102, 2012.

[9] M. Soysal and E. G. Schmidt, "Machine learning algorithms for accurate flow-based network traffic classification: evaluation and comparison," *Performance Evaluation*, vol. 67, no. 6, pp. 451–467, 2010.

[10] Y. Liu, W. Li, and Y. Li, "Network traffic classification using K-means clustering," in *Proceedings of the 2nd International Multi-Symposiums on Computer and Computational Sciences (IMSCCS)*, pp. 360–365, Iowa City, IA, USA, August 2007.

[11] A. Shrivastav and A. Tiwari, "Network traffic classification using semi-supervised approach," in *Proceedings of the 2nd International Conference on Machine Learning and Computing (LCMLC)*, pp. 345–349, Bangalore, India, February 2010.

[12] P. Teufl, U. Payer, M. Amling et al., "InFeCT - network traffic classification," in *Proceedings of the 7th International Conference on Networking (ICN)*, pp. 439–444, Cancun, Mexico, April 2008.

[13] D. Bekerman, B. Shapira, L. Rokach, and A. Bar, "Unknown malware detection using network traffic classification," in *Proceedings of the 2015 IEEE Conference on Communications and Network Security (CNS)*, pp. 134–142, Florence, Italy, September 2015.

[14] X. Mu and W. Wu, "A parallelized network traffic classification based on hidden Markov model," in *Proceedings of the 2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp. 107–112, Beijing, China, October 2011.

[15] P. C. Sethi and P. K. Behera, "Internet traffic classification for faster and secured network service," *International Journal of Computer Applications*, vol. 131, no. 4, pp. 15–20, 2015.

[16] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: an overview," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 76–81, 2019.

[17] H. Lim, J. Kim, J. Heo, K. Kim, Y. Hong, and Y. Han, "Packet-based network traffic classification using deep learning," in *Proceedings of the 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pp. 46–51, Okinawa, Japan, February 2019.

[18] "NetFlow," 2020, https://www.manageengine.com/products/netflow/.

[19] D. Harrington, R. Presuhn, and B. Wijnen, "An architecture for describing simple network management protocol (SNMP) management frameworks," *RFC3411, STD*, vol. 62, 2002.

[20] W. Guo, D. Y. Yao, Y. Fu et al., "Study on the method for regional traffic flow feature extraction and traffic status evaluation," *Journal of Highway and Transportation Research and Development*, vol. 22, no. 7, pp. 101–104, 2005.

[21] S. Lei, "A feature selection method based on information gain and genetic algorithm," in *Proceedings of the 2012 International Conference on Computer Science and Electronics Engineering(ICCSEE)*, vol. 2, pp. 355–358, Hangzhou, China, March 2012.

[22] H. Kaindl and G. Kainz, "Bidirectional heuristic search reconsidered," *Journal of Artificial Intelligence Research*, vol. 7, pp. 283–317, 1997.

[23] S. Maldonado and R. Weber, "A wrapper method for feature selection using Support Vector Machines," *Information Sciences*, vol. 179, no. 13, pp. 2208–2217, 2009.

[24] H. A. Fayed and A. F. Atiya, "Speed up grid-search for parameter selection of support vector machines," *Applied Soft Computing*, vol. 80, pp. 202–210, 2019.

[25] C. F. Lin and S. D. Wang, "Fuzzy support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 464–471, 2002.

[26] A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, no. 2003, pp. 1157–1182, 2003.

[27] B. Scholkopf, K. K. Kah-Kay Sung, C. J. C. Burges et al., "Comparing support vector machines with Gaussian kernels to radial basis function classifiers," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2758–2765, 1997.

[28] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, no. 2, pp. 65–85, 2014.

[29] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, vol. 5, pp. 4104–4108, Orlando, FL, USA, October 1997.

[30] M. Neshat, G. Sepidnam, M. Sargolzaei, and A. N. Toosi, "Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications," *Artificial Intelligence Review*, vol. 42, no. 4, pp. 965–997, 2012.

[31] J. H. Sun, T. H. Jeng, C. C. Chen, H. C. Huang, and K. S. Chou, "MD-Miner: behavior-based tracking of network traffic for malware-control domain detection," in *Proceedings of the IEEE 3rd International Conference on Big Data Computing Service and Applications (BigDataService)*, pp. 96–105, San Francisco, CA, USA, April 2017.

[32] J. Mercer, *Functions Ofpositive and Negativetypeand Their-commection with the Theory Ofintegral Equations*, pp. 4–415, Philosophicol Trinsdictions of the Rogyal Society, London, UK, 1909.

[33] J. Caida, https://www.caida.org/home/, 2020.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 6638134, 13 pages
https://doi.org/10.1155/2021/6638134

Research Article

Representativeness-Based Instance Selection for Intrusion Detection

Fei Zhao,[1] **Yang Xin**,[1,2] **Kai Zhang**,[1] **and Xinxin Niu**[1,2]

[1]*National Engineering Laboratory for Disaster Backup and Recovery, Information Security Center, School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2]*Guizhou Provincial Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China*

Correspondence should be addressed to Yang Xin; yangxin@bupt.edu.cn

Received 26 November 2020; Revised 24 January 2021; Accepted 26 February 2021; Published 13 March 2021

Academic Editor: Entao Luo

With the continuous development of network technology, an intrusion detection system needs to face detection efficiency and storage requirement when dealing with large data. A reasonable way of alleviating this problem is instance selection, which can reduce the storage space and improve intrusion detection efficiency by selecting representative instances. An instance is representative not only in its class but also in different classes. This representativeness reflects the importance of an instance. Since the existing instance selection algorithm does not take into account the above situations, some selected instances are redundant and some important instances are removed, increasing storage space and reducing efficiency. Therefore, a new representativeness of instance is proposed and considers not only the influence of all instances of the same class on the selected instance but also the influence of instances of different classes on the selected instance. Moreover, it considers the influence of instances of different classes as an advantageous factor. Based on this representativeness, two instance selection algorithms are proposed to handle balanced and imbalanced data problems for intrusion detection. One is a representative-based instance selection for balanced data, which is named RBIS and selects the same proportion of instances from each class. The other is a representative-based instance selection for imbalanced data, which is named RBIS-IM and selects important majority instances according to the number of instances of the minority class. Compared with other algorithms on the benchmark data sets of intrusion detection, experimental results verify the effectiveness of the proposed RBIS and RBIS-IM algorithms and demonstrate that the proposed algorithms can achieve a better balance between accuracy and reduction rate or between balanced accuracy and reduction rate.

1. Introduction

Along with the continuous development of network technology and 5G, smart systems are becoming more and more common in all fields of human life, such as finance, agriculture, and education. However, smart systems have become the target of many new attacks, which not only cause significant financial damage and personal information leakage but also hinder the large-scale deployment of smart systems in practice. As intrusion detection technology can effectively protect smart systems and detect attacks, the development of intrusion detection technology has attracted the attention of countries all over the world [1, 2]. From the perspective of classification, the main goal of building an intrusion detection system (IDS) is to train a classifier that can distinguish between normal and intrusive data from the original network data set.

The IDS based on machine learning has become an important part of IDS [3], which directly uses a large amount of network data to detect attacks. These network data can result in wasting time and storage space for IDS. Moreover, the redundant data and noise in these data can affect the performance of IDS. But, instance selection is used for IDS to select important data from the original data to achieve two goals. One is to reduce the number of instances required by IDS in the training phase, thereby saving time and reducing the amount of calculation for training the classifier; the other is that through effective instances, the performance of the trained classifier can be effectively improved [4–6].

In recent years, many instance selection techniques have been proposed to improve the performance of IDS [7–15]. However, in terms of the factors and application areas of instance selection, there are mainly four problems in existing instance selection algorithms.

Firstly, only the influence of a portion of instances is taken into account for selecting the instance [7–9]. For instance, the instance selection algorithm based on partition and cluster center (PCIS) [7] selects representative instance by considering K nearest neighbor instances of the same class; The binary nearest neighbor tree algorithm (BNNT) [8] and the constraint nearest neighbor-based instance reduction algorithm (CNNIR) [9] select representative instance by K nearest neighbor instances of the selected instance. As these instance selection algorithms only consider the influence of a portion of instances and ignore the influence of remaining instances, some important instances are not selected.

Secondly, the influence of instances of different classes is regarded as an adverse factor selecting the instance. For the instances of the same class, the instance selection algorithm based on a ranking procedure (ISAR) [10] and the ranking-based instance selection algorithm (RIS) [11] only select instances representing the same class and remove instances representing different classes. Some selected instances are not representative because the influence of instances of different classes is considered as an adverse factor.

Thirdly, some instance selection algorithms use sampling to select the instance. Instance selection algorithm based on hierarchical data topology [12] uses hierarchical sampling to deal with large-scale problems of data sets. This algorithm combines the random subset selection (RSS) with the topology-based selection (TBS) to select important instances, which is a subset of original instances. Since sampling is used to select instance, some important instances are still removed, which contain the information of or original instances.

Finally, in the field of intrusion detection, only a few algorithms are used to deal with imbalanced data, and most of the instance selection algorithms are used to deal with the problem of balanced data [13–15]. Data imbalance is known as instance imbalance. For the binary classification problem, under normal circumstances, the proportion of positive and negative instances should be relatively close, and many existing classification models are based on this assumption. However, in some specific scenarios, the proportion of positive and negative instances may vary greatly, which reduces the accuracy of minority class, which has smaller instances. Therefore, instance selection algorithms, which deal with imbalanced data, need to be strengthened.

Given the above four problems, the factors considered for instance selection include: (1) the influence of all instances of the same class on the selected instance; (2) the influence of instances of different classes on the selected instance; (3) the influence of different classes of instances as an advantageous factor; and (4) the instance selection algorithm should be applied to the balanced and unbalanced domains for intrusion detection. As the existing instance selection algorithm does not take into account the above four factors, some selected instances are redundant and some important instances are removed, increasing storage space and reducing efficiency. Therefore, for the first three factors, we propose a new concept of representativeness of instance. This concept is used to express the importance of the instance. Considering the fourth factor, we propose two representativeness-based instance selections, which are named RBIS and RBIS-IM. RBIS algorithm is used to handle balanced data and select the same proportion of instances from each class. And RBIS-IM algorithm is used to deal with imbalanced data and select important majority instances according to the number of instances of the minority class. Finally, the experimental results verify the effectiveness of proposed algorithms. Two algorithms can reduce the size of the training set while maintaining or even increasing accuracy (ACC) and balanced accuracy (BA).

The main contributions of this paper are as follows:

(1) A new concept of instance representativeness is proposed to represent the importance of an instance. In terms of instance representativeness, we consider not only the representativeness of the instance within its class but also the representativeness of the instance within different classes. The two representativenesses are advantageous factors;

(2) To deal with balanced data problem, the RBIS algorithm, which is based on instance representativeness, is designed to select the same proportion of normal instances and attack instances to improve intrusion detection efficiency. Compared with other algorithms on the benchmark data sets of intrusion detection, RBIS algorithm can achieve a better balance between accuracy and reduction rate.

(3) To handle imbalanced data problem, the RBIS-IM algorithm, which is based on instance representativeness, is designed to select the same number of normal instances and attack instances. Compared with other algorithms on the benchmark data sets of intrusion detection, RBIS-IM algorithm can achieve a better balance between balanced accuracy and reduction rate.

The paper is structured as follows. In Section 2, we introduce the basic concepts of instance selection technique. Section 3 reports a new concept of instance representativeness and two representativeness-based instance selection algorithms that are used with regard to balanced and imbalanced problems, respectively. Experimental results with two representative-based instance selection algorithms are shown in Section 4. Finally, conclusions with a discussion on future work are presented in Section 5.

2. Instance Selection Technique

In this section, the basic concepts of the instance selection technique are introduced. The instance selection is to select important instances and eliminate redundant instances from the original data. These selected instances can contain the total effective information of the original data. Suppose X

represents the original data; S represents the selected instances; so S is the subset of X, i.e., $S \subset X$ and $|S| \ll |X|$ .Using the instance subset S for IDS can improve detection efficiency and reduce storage requirements. According to the distribution of instances and selection strategies, instances with different locations play different roles in the classification process. In general, these algorithms are divided into three categories: condensation, edition, and hybrid.

The condensation algorithm considers that instances close to the boundary play an important role in the classification process, just like SVM. It preserves the boundary instances by deleting the interior instances of each class [16–18]. In the field of intrusion detection, nature-inspired instance selection technique (NIIS) [19] and instance selection technique based on cuckoo search and bat algorithm (CSBAIS) [20] are proposed to improve the training speed and accuracy of the support vector machine (SVM). The NIIS algorithm applies the lower polling algorithm and social spider algorithm to select instances near the boundary. CSBAIS algorithm uses cuckoo search and bat algorithm to select instances near the boundary. But these algorithms remove some important internal instances too.

The edition algorithm is the opposite of the condensation algorithm. It tends to smooth the class boundary by deleting the boundary instances [21–23]. The instance selection algorithm based on K-means and K-nearest neighbor (KMKNNIS) [24] is proposed to select important internal instances. Those instances near the boundary are removed. The penalty-reward-based instance selection method [25] is to select instances by removing noise and boundary instances. These algorithms can ignore some critical boundary instances.

Finally, the hybrid algorithm combines the condensation algorithm with the edition algorithm to obtain a smaller subset and an acceptable accuracy in the testing set [9, 26–28]. PCIS [7] algorithm applies the partition and cluster center to select the instance. First, the algorithm only considers the influence of k instances of the same class on the selected instances and does not consider the influence of all instances of the same class. Second, the algorithm only uses the class center instances of different classes and does not use the information of all instances of different classes. Third, the instance information of different classes is regarded as adverse information. ISAR [10] and RIS [11] algorithms select important instances by sorting the instances. In the process of sorting instances, although the influence of all instances of different classes is considered, it is regarded as adverse information. BNNT algorithm uses the binary nearest neighbor tree to select the instance [8]. The algorithm only considers the k nearest neighbor instances of the selected instance and does not consider the influence of remaining instances. Moreover, the algorithm needs to delete internal instances to select instances. The CNNIR algorithm uses the constraint nearest neighbor to select the instance [9]. The algorithm does not consider the influence of remaining instances.

To sum up, there are mainly four factors in the instance selection process: (1) the influence of all instances of the same class on the selected instance; (2) the influence of instances of different classes on the selected instance; (3) the influence of different classes of instances as an advantageous factor; and (4) the instance selection algorithm should be applied to the balanced and imbalanced domains for intrusion detection. Since the above four factors are not taken into account in the existing instance selection algorithm, some selected instances are redundant and some important instances are removed, increasing storage space and reducing efficiency. Therefore, we propose two algorithms to select important instances without deleting internal instances, which can handle balanced and imbalanced data problems. Meanwhile, the proposed algorithms consider not only the influence of all instances of the same class on the selected instances but also the influence of instances of different classes and take the influence of instances of different classes as an advantageous factor.

3. Proposed Algorithms

In this section, we introduce the proposed representativeness-based instance selection algorithms. In the first subsection, we introduce a new instance representativeness. In the next two subsections, two representativeness-based algorithms are introduced, which are used to deal with balanced and imbalanced data problems.

3.1. Proposed Instance Representativeness.

The key factor of instance selection is to decide which instance is representative, which makes the selected instance subset representativeness of the original data. Selecting representative instances, we should consider not only the representativeness of the selected instance category but also the representativeness of different categories. In other words, the instance selected has the information of its category and different categories. And the influence of instances of different categories is seen as an advantageous factor.

Suppose that X is a training instance set containing normal and attack categories, $X = \{(x_1, c_1), \ldots, (x_n, c_2)\}$. X has n instances; x_i is a d-dimensional instance; c expresses the classes of instances and $c = \{c_1, c_2\}$; c_1 is the class of normal instances X_n and c_2 is the class of attack instances X_a; X is composed of X_n and X_a.

The representation of any instance x_i in the training set X is as follows:

$$R(x_i, c) = \{Q(x_i, c_r)\} * \{Q(x_i, c_p)\}. \quad (1)$$

The first half of formula (1) represents the representativeness of instance x_i in its category; the second half shows the representativeness of instance x_i in different categories; $c_r, c_p \subset \{c_1, c_2\}$ and $r \neq p$; c_r represents the category of instance x_i; c_p is a different category from instance x_i.

To realize $Q(x_i, c_r)$ or $Q(x_i, c_p)$ in formula (1), the Euclidean distance $d(x_i, x_j)$ can be used to represent the relation of two instances. The representativeness between an instance and a class is inversely proportional to the sum of its Euclidean distances of the instance and remaining instances of the same class. And the representativeness of instances of different categories is considered.

Thus, formula (1) is transformed into the following form:

$$R(x_i, c) = \left\{ \frac{1}{\left(\sum_{c_i = c_j, j=1}^{n_i} d(x_i, x_j) \right)} \right\} * \left\{ \frac{1}{\sum_{c_i \neq c_j, j=1}^{n_j} d(x_i, x_j)} \right\}, \tag{2}$$

where n_i is the number of instances in the same category as x_i; n_j is the number of instances in a different category from x_i. The expression c_i shows the category of instance x_i; the expression $c_i = c_j$ shows that instances x_i and x_j are the same category; the expression $c_i \neq c_j$ shows that instances x_i and x_j are different categories; if x_i and x_j are the same class, $i \neq j$.

Calculating the representativeness of instance $R(x_i, c)$, three factors are considered: (1) the influence of all instances of the same class on the selected instance; (2) the influence of instances of different classes on the selected instance; and (3) the influence of different classes of instances as an advantageous factor. The proposed representativeness of instance reflects the importance of instance. In Section 4.3, compared with other algorithms on the benchmark data sets of intrusion detection, experimental results verify the effectiveness of the representativeness of instance $R(x_i, c)$.

3.2. Representativeness-Based Instance Selection for Balanced Data. To handle balanced data problem, a representativeness-based instance selection algorithm is proposed to select representative instances, which is called RBIS, to improve accuracy (ACC) and reduce reduction rate (RR) for IDS. Through the RBIS algorithm, the same proportion of instances for each class is selected. Algorithm 1 shows the pseudo-code of the RBIS algorithm.

In Algorithm 1, original instances X are composed of normal instances X_n and attack instances X_a.S is the set of selected instances from original instances X;S_n is the set of selected normal instances from X; S_a is the set of selected attack instances from X; the parameter t is the ratio of selected instances by cross-validation or validation set. Firstly, in lines 3–5 of Algorithm 1, the representativeness of each instance is calculated. According to normal instances X_n and attack instances X_a, S_n and S_a are initialized. Secondly, according to representativeness $R(x_i, c)$, representativeness $R(x_i, c)$ and training set X are sorted in descending order (lines 6 and 7). Meanwhile, S_n and S_a are sorted in descending order. Thirdly, from line 8 to line 11, according to the cross-validation or validation data, 1-NN is used as the classifier. The parameter t with the best accuracy is selected and the range of parameter t is [0, 1]. In Section 4.3, the selection process of parameter$t$ is shown by Figures 1 and 2. According to parameter t, the first $|S_n| * t$ instances and the first $|S_a| * t$ instances are selected in S_n and S_a, respectively. Finally, according to S_n and S_a, S is determined.

Figure 3 with two dimensions is used to demonstrate the instance selection process of the RBIS algorithm. Figure 3(a) shows two types of original data, which are normal and attack instances. The circle is "Class One," which represents the normal instance; the square is "Class Two," which represents the attack instance. And there are 10 normal instances and 10 attack instances. According to their representativeness, the instances of each class are ranked in Figure 3(b). The numbers around the graph indicate the degree of representation of the instance. The smaller the number, the more representative the instance is. For example, in normal instances, the Number "1" is the most representative and the Number "10" is the least representative. In Figure 3(c), according to the parameter t, the same proportion of instances are selected in each class. When the parameter t is 0.6, the first six instances of each class are selected.

The RBIS algorithm is based on the representativeness $R(x_i, c)$ of instance. The selected instances of RBIS algorithm contain the information of original data. The efficiency of the RBIS algorithm is related to the accuracy (ACC) and reduction rate (RR). Compared with other algorithms on the benchmark data sets of intrusion detection, experimental results, which are shown in Section 4.3, prove that the RBIS algorithm is effective and achieves a better balance between accuracy and reduction rate. As the same proportion of instances for each class is selected, the RBIS algorithm can handle the balanced data problem.

According to Algorithm 1 and formula (2), the time complexity of the proposed algorithm is mainly related to the calculation of instance distance between the same and different classes. Therefore, the time complexity of the algorithm is $O(N^2) + O(M)$, where N represents the total number of training instances and M represents the number of experiments conducted by the classifier when selecting the parameter t. As $O(M)$ is far less than $O(N^2)$, the time complexity of RBIS is $O(N^2)$.

3.3. Representativeness-Based Instance Selection for Imbalanced Data. To solve the imbalanced data problem, a representativeness-based instance selection algorithm is proposed, which is called RBIS-IM. Through the RBIS-IM algorithm, the same number of instances for each class is selected to improve balanced accuracy (BA) and reduce reduction rate (RR) for IDS.

Algorithm 2 shows the pseudo-code of the RBIS-IM algorithm. Like Algorithm 1, Algorithm 2 is based on the representativeness of instance. Original instances X are composed of normal instances X_n and attack instances X_a. X_n and X_a are called the majority class and the minority class, respectively. The difference in the number between X_n and X_a is huge. S is the set of selected instances from original instances X; S_n is the set of selected normal instances from X; S_a is the set of selected attack instances from X; the parameter t is the ratio of selected instances by cross-validation or validation set.

In the process of instance selection, the number of selected instances of the majority class not only depends on the number of instances of the minority class but also is the same as that selected of the minority class. Firstly, in lines 3–5 of Algorithm 2, the representativeness of each instance is calculated. According to X_n and X_a, S_n and S_a are initialized. Secondly, according to representativeness,

```
Input: $X$: Training data set; $t$: the Ratio of selected instance by cross-validation or validation set; $X_n$: the Set of normal instances; $X_a$: the Set of attack instances.
Output: $S = S_n \cup S_a$; $S$: Set of selected instances from $X$; $S_n$: Set of selected normal instances from $X_n$; $S_a$: Set of selected attack instances from $X_a$
(1) Normalize $X$
(2) Initialize $S$,$S_n$, and $S_a$, according to $X$, $X_n$, and $X_a$
(3) For each $x_i$ in $X$
(4) calculate $R(x_i, c)$ by formula (2)
(5) End for
(6) $[R(x_i, c), I] \longleftarrow \text{sortdesc}\{R(x_i, c)\}$
(7) $X \longleftarrow \text{sortIdx}(X, I)$
(8) Obtain $S_n$ and $S_a$; in other words, according to $R(x_i, c)$, $S_n$ and $S_a$ are sorted in descending order
(9) Select the best $t$ that reaches the best accuracy using 1-NN classifier through cross validation or validation set
(10) Obtain $S_n \longleftarrow S_n * t$ and $S_a \longleftarrow S_a * t$, which select the first $|S_n| * t$ instances in $S_n$ and the first $|S_a| * t$ instances in $S_a$
(11) Obtain $S \longleftarrow S_n \cup S_a$
```

Algorithm 1: RBIS.

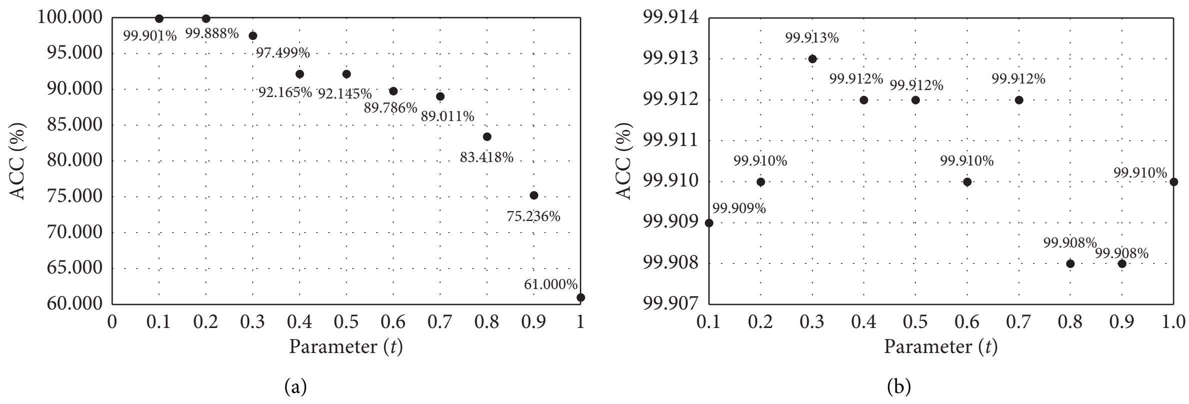


Figure 1: The relation of ACC and parameter t on the DoS data set. (a) $t = [0.1, 0.2, \ldots, 1]$; (b) $t = [0.001, 0.002, \ldots, 0.01]$.

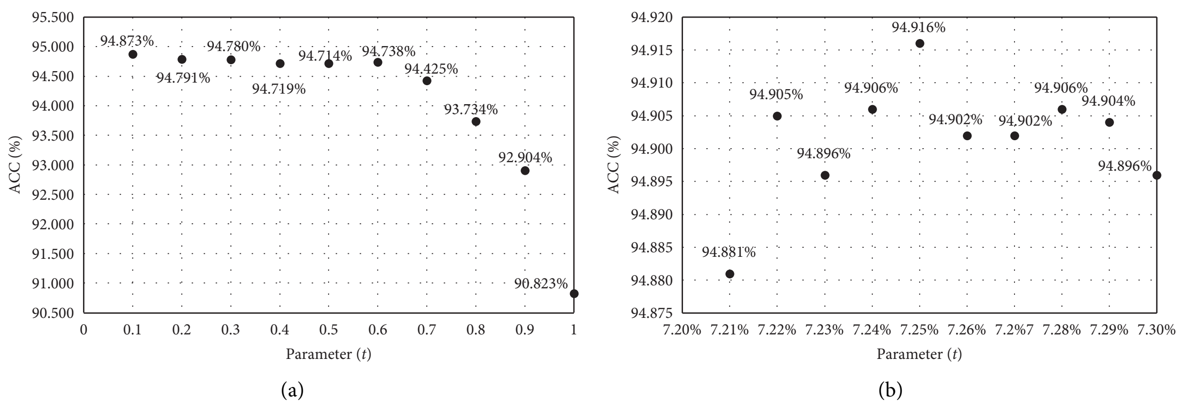


Figure 2: The relation of ACC and parameter t on the DDoS 2016 data set. (a) $t = [0.1, 0.2, \ldots, 1]$; (b) $t = [0.0721, 0.0722, \ldots, 0.0730]$.

representativeness $R(x_i, c)$ and training set X are sorted in descending order (lines 6 and 7). Meanwhile, S_n and S_a are also sorted in descending order. Thirdly, from line 8 to line 11, according to the cross-validation or validation data, 1-NN is used as the classifier; the parameter t with the best balanced accuracy (BA) is selected and the range of parameter t is [0, 1]. In Section 4.3, the selection process of parameter t is shown by Figures 4–6. According to the selected parameter t, select the first $|S_a| * t$ instances and the first $|S_a| * t$ instances from in S_n and S_a, respectively. Finally, according to S_n and S_a, S is determined.

Figure 7 with two dimensions is used to explain the instance selection process of the RBIS-IM algorithm. Figure 7(a) shows two types of original data where the circle

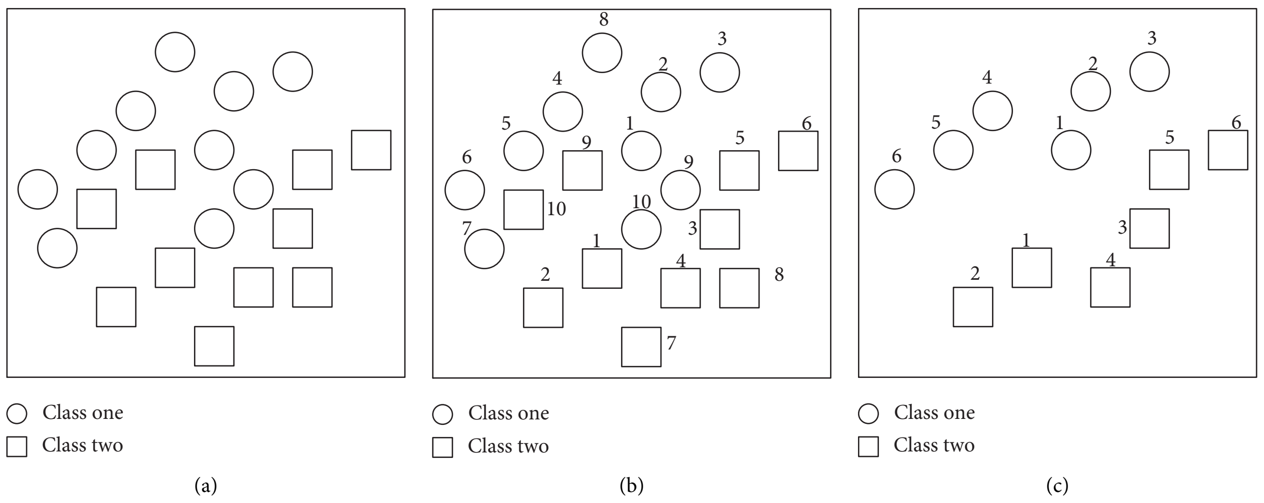


FIGURE 3: RBIS selects the critical instances of all classes in the balanced data. (a) Original data. (b) Sort the instances according to their representativeness. (c) Select the same proportion of instances according to the parameter (t).

```
Input: $X$: Training data set; $t$: the Ratio of selected instance by cross-validation or validation set; $X_n$: the Set of normal instances called the majority class; $X_a$: the Set of attack instances called the minority class.
Output: $S = S_n \cup S_a$; $S$: Set of selected instances from $X$; $S_n$: Set of selected normal instances from $X_n$; $S_a$: Set of selected attack instances from $X_a$
(1) Normalize $X$
(2) Initialize $S$,$S_n$, and $S_a$, according to $X$, $X_n$, and $X_a$
(3) For each $x_i$ in $X$
(4) Calculate $R(x_i, c)$ by formula (2)
(5) End for
(6) $[R(x_i, c), I] \longleftarrow \text{sortdesc}\{R(x_i, c)\}$
(7) $X \longleftarrow \text{sortIdx}(X, I)$
(8) Obtain $S_n$ and $S_a$; In other words, according to $R(x_i, c)$, $S_n$ and $S_a$ are sorted in descending order.
(9) Select the best $t$ that reaches the best balanced accuracy using 1-NN classifier through cross-validation or validation set
(10) Obtain $S_a \longleftarrow S_a * t$ and $S_n \longleftarrow S_a * t$, which select the first $|S_a| * t$ instances in $S_n$ and the first $|S_a| * t$ instances in $S_a$
(11) Obtain $S \longleftarrow S_n \cup S_a$
```

ALGORITHM 2: RBIS-IM.

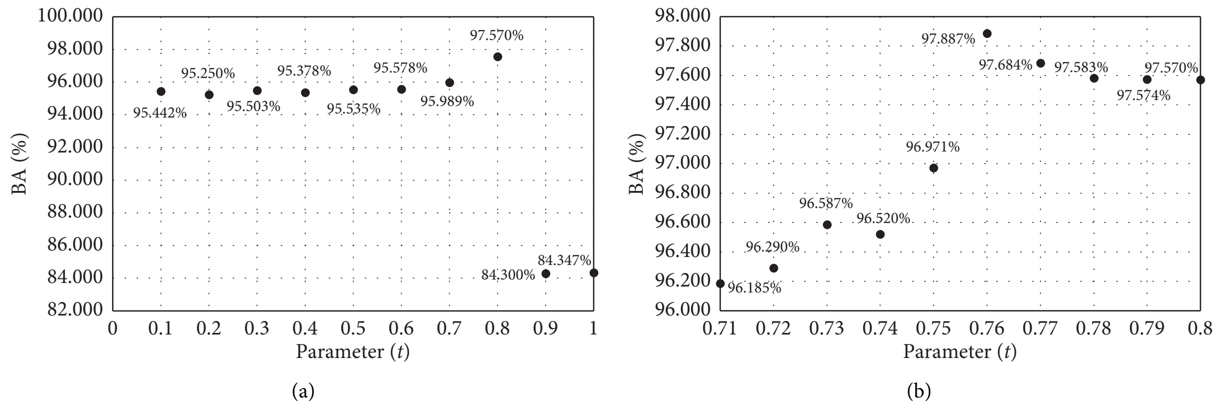


FIGURE 4: The relation of BA and parameter t on the Probe data set. (a) $t = [0.1, 0.2, \ldots, 1]$; (b) $t = [0.71, 0.72, \ldots, 0.80]$.

expresses majority class and the square expresses minority class. There are 8 instances in majority class, and there are 4 instances in minority class. According to their representativeness, the instances of each class are ranked in Figure 7(b). Similarly, the numbers around the graph indicate the degree of representation of the instance. The

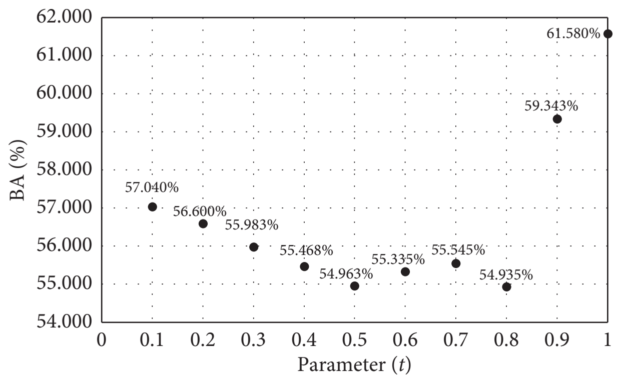


Figure 5: The relation of BA and parameter t on the U2R data set.

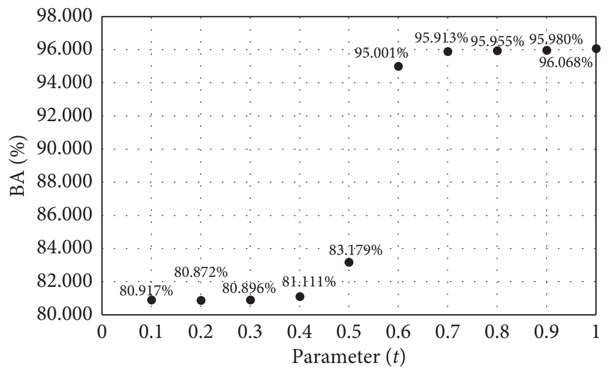


Figure 6: The relation of BA and parameter t on the R2L data set.

smaller the number, the more representative the instance is. In Figure 7(c), when the parameter t is 1, the first four instances of the minority class are selected. Since the number of selected instances of the majority class depends on the number of instances of the minority class and is the same as that selected of the minority class, the first four instances of the majority class are also selected.

Similarly, since the RBIS-IM algorithm is based on the representativeness of instance $R(x_i, c)$, the selected instances can contain all information of original data. And the effectiveness of RBIS-IM algorithm is evaluated by balanced accuracy (BA) and reduction rate (RR). In Section 4.3, compared with other algorithms on the benchmark data sets of intrusion detection, experimental results show that the RBIS-IM algorithm is effective and can achieve a better balance between BA and RR. Since the same number of instances for each class is selected to improve intrusion detection efficiency, RBIS-IM algorithm can deal with the imbalanced data problem. As the time complexity of the RBIS-IM algorithm is the same as the RBIS algorithm, the time complexity of this algorithm is $O(N^2)$.

The difference between RBIS-IM and RBIS algorithms is mainly embodied in three aspects. Firstly, the problems solved by the two algorithms are different. The RBIS-IM algorithm is to solve imbalanced data problem, which refers to the huge difference in the number of normal instances and attack instances; the RBIS algorithm is to deal with balanced data problem, which means that the number of normal instances and attack instances is very close or equal. Secondly, the methods of selected instances of two algorithms are different. In the RBIS-IM algorithm, the selection of instances of majority class is determined by selected instances of minority class. The number of selected instances of two classes is the same. In the RBIS algorithm, the number of instances of each class is close. In the RBIS algorithm, the same proportion of instances are selected for each class. Therefore, the number of selected normal and attack instances is very close. Thirdly, the evaluation criteria of the two algorithms are different, which are shown in Section 4.2. RBIS is evaluated by ACC and RR while RBIS-IM is related to BA and RR.

4. Experiments

In this section, experiments are designed to prove the effectiveness of the proposed algorithms. The section is divided into three subsections. In the first subsection, two experimental data sets are shown. In the second subsection, the evaluation criteria are introduced. In the last subsection, the RBIS and RBIS-IM algorithms are validated on balanced and imbalanced data sets.

4.1. Experimental Data Set. In this article, we use two data sets, which are the Knowledge Discovery and Data Mining

(KDD) Cup 1999 data set and DDoS 2016 data set. Although the KDD 99 data set has some disadvantages, it is still widely used as a benchmark for IDS evaluation [29–31]. In the KDD 99 data set, the 10% KDD training data and the KDD correct data are used as training data and testing data, respectively. The distribution of these data is shown in Table 1. In the KDD Cup 99 data set, the label of data includes the normal class and attack classes, which are divided into four groups: the remote-to-login (R2L), the denial-of-service (DoS), the user-to-root (U2R), and the Probe.

In the KDD Cup 99 data set, every network connection represents a data record that consists of 41 features and a label specifying the status of this record. Each record contains 41 features: 3 nonnumeric features, and 38 numeric features. During data preprocessing, these nonnumeric features, which are the protocol type, service, and flag, must be transformed into numeric data. The protocol type has three kinds of types: tcp, udp, and icmp. According to the different types, the "protocol type" feature is transformed into three features. As the "service" feature has 70 different types and would heavily increase the dimensionality, this single feature is not used in our experiments. The nonnumeric feature conversion is shown in Table 2.

The DDoS 2016 data set was published in 2016, which was created using the network simulator NS2 [32, 33]. There are 2.1 million data records in the data set. Each record contains 28 features: 5 nonnumeric features, and 23 numeric features. These nonnumeric features need to be converted to numerical ones. The data set contains normal data and four types of DDoS attacks, which are UDP flood, smurf, HTTP flood, and SIDDOS. In this section, the data set, which uses normal data and UDP flood, is used to evaluate the performance of the proposed algorithms.

According to balanced and imbalanced domains, the Knowledge Discovery and Data Mining (KDD) Cup 1999 and DDoS 2016 are divided into the balanced data set and the imbalanced data set. The description of data sets is shown in Tables 3 and 4.

Table 1: The distribution of the KDD 99 data.

Class	The 10% KDD training data	The KDD correct data
Normal	97278	60593
DoS	391458	229853
U2R	52	228
Probe	4107	4166
R2L	1126	16189
Total	494021	311029

Table 2: The nonnumeric feature conversion in the KDD 99 data.

Feature name	Type setting 1	Type setting 2
Protocol type = tcp	tcp = 1	others = 0
Protocol type = udp	udp = 1	others = 0
Protocol type = icmp	icmp = 1	others = 0
Flag	SF = 1	others = 0

Table 3: The balanced data set.

Type	Attribute	Class	Normal/ attack in training data	Normal/ attack in testing data
Normal and DoS data in the KDD 99 data set	42	2	10000/10000	10000/10000
The DDoS 2016 data set	28	2	10000/10000	10000/10000

Table 4: The imbalanced data set.

In the KDD 99 data set	Attribute	Class	Normal/attack in training data	Normal/attack in testing data
Normal and U2R data	42	2	10000/30	200/20
Normal and probe data	42	2	10000/1550	10000/1000
Normal and R2L data	42	2	10000/1000	10000/1000

4.2. Evaluation Criteria. To evaluate the effectiveness and performance of the proposed algorithms, the confusion matrix is used. The confusion matrix is shown in Table 5. According to the confusion matrix, four performance metrics are applied: the detection rate (DR, also known as the true positive rate), true negative rate (TNR, also known as specificity or selectivity), balanced accuracy (BA), and accuracy (ACC). Meanwhile, the reduction rate (RR) is also applied.

Table 5: Confusion matrix.

Class	Predicted negative class	Predicted positive class
Actual negative class	True negative (TN)	False positive (FP)
Actual positive class	False negative (FN)	True positive (TP)

In balanced data, ACC and RR are used to evaluate the performance of the proposed RBIS algorithm. To treat the minority and majority instances equally, BA is selected as the evaluation criterion of the RBIS-IM algorithm in the imbalanced problem.

The DR is the proportion of attack instances that are correctly predicted as attacks in the test data set; it is an important metric reflecting the attack detection model's ability to identify attack instances and is described as

$$\mathrm{DR} = \frac{\mathrm{TP}}{P} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}. \tag{3}$$

The TNR is the proportion of normal instances that are correctly predicted as normal in the test data set. And, it is an important metric reflecting the detection model's ability to identify normal instances and can be written as

$$\mathrm{TNR} = \frac{\mathrm{TN}}{N} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}. \tag{4}$$

The BA is the average of DR and TNR; it can be a leading metric for imbalanced data sets; it can serve as an overall performance metric for a model.

$$\mathrm{BA} = \frac{\mathrm{DR} + \mathrm{TNR}}{2}. \quad (5)$$

The ACC is the ratio of the number of instances correctly predicted in the test data set to the total number of instances. And, it can reflect the ability of the detection model to distinguish between normal and attack instances and is defined as

$$\mathrm{ACC} = \frac{\mathrm{TN} + \mathrm{TP}}{P + N} = \frac{\mathrm{TN} + \mathrm{TP}}{\mathrm{TN} + \mathrm{TP} + \mathrm{FN} + \mathrm{FP}}. \quad (6)$$

The RR is the ratio of the number of selected instances in the training data set to the total number of instances; it can show the ability of the instance selection model to select optimal instances and can be written as

$$\mathrm{RR} = \frac{|S|}{\|X\|} * 100\%. \quad (7)$$

4.3. Experimental Results and Analysis. In this section, we use the instance subset selected by the proposed instance selection algorithms to verify the effectiveness of instance representation and the algorithms. The experiment is conducted in balanced and imbalanced data sets. All the experimental results are obtained by calculating the average value of 100 experiments.

The RBIS and RBIS-IM algorithms have a parameter t that is used to determine the number of selected instance subsets. In the training phase, the parameter t is determined by grid search on cross validation or verification set. In the RBIS algorithm, the parameter is selected by the best ACC. In the RBIS-IM algorithm, the selected parameter is related to the best BA.

Figures 1 and 2 show the relation of ACC and parameter t on the balanced data sets. Moreover, Figures 1 and 2 reflect the selection process of parameter t in RBIS algorithm on DOS and DDOS 2016 data sets. Figures 1(a) and 2(a) display the change of ACC when the parameter t is in a large interval [0.1, 1]. Figures 1(b) and 2(b) show the change of ACC when the parameter t is in a small interval [0.001, 0.01] and [0.0721, 0.0730]. Figure 1(b) is based on Figure 1(a). Similarly, Figure 2(b) is based on Figure 2(a). From Figure 1(a), the best ACC is achieved when the parameter t takes 0.1 in the interval [0.1, 1]. Therefore, the range of parameter t in Figure 1(b) is in the interval [0, 0.1]. Through experiments, the range of parameter t in Figure 1(b) is in the interval [0.001, 0.01]. In Figure 1(b), according to the best ACC, the parameter t is 0.3%.

Like Figure 1, Figure 2(a) illustrates that the best ACC is obtained when the parameter t takes 0.1 in the interval [0.1, 1].Therefore, the range of parameter t in Figure 2(b) is in the interval [0, 0.1]. Through experiments, the range of parameter t in Figure 2(b) is in the interval [0.0721, 0.0730]. In Figure 2(b), according to the best ACC, the parameter t is 7.25%.

Figures 4-6 display the relation of BA and parameter t on the imbalanced data sets. When the parameter t is in the interval [0.1, 1] and [0.71, 0.80], the change of BA on the Probe data set is shown in Figures 4(a) and 4(b). Figures 5 and 6 show BA changes on the U2R and R2L data sets when the parameter t is in the interval [0.1, 1]. Meanwhile, Figures 4–6 reflect the selection process of parameter t in the RBIS-IM algorithm. Figures 4(a) show the change of BA when the parameter t is in a large interval [0.1, 1]. Figures 4(b) indicate the change of BA when the parameter t is between in a small interval [0.71, 0.80]. Figure 4(b) is based on Figure 4(a). From Figure 4(a), the best BA is obtained when the parameter t takes 0.8 in the interval [0.1, 1].Therefore, the range of parameter t in Figure 4(b) is in the interval [0, 0.8]. Through experiments, the range of parameter t in Figure 4(b) is in the interval [0.71, 0.80]. In Figure 4(b), according to the best BA, the parameter t is 0.76. From Figures 5 and 6, it is obvious that the parameter t is set to 1 under the condition that BA obtains the best on two data sets. Moreover, relevant experiments are conducted in the interval [0.9, 1]. The experimental results show that BA obtains the best when parameter t is 1.

Table 6 shows that on the balanced data set, the three common classifiers, which are 1-NN, SVM, and Adaboost, use the entire training set and instance subset selected to obtain ACC, RR, and average accuracy, respectively. On the DoS data set of KDD cup 99, the accuracy of the three classifiers is greatly improved by using the instance subset selected by the RBIS algorithm. On the DDoS 2016 data set, the three classifiers also achieve good accuracy by using the instance subset. The accuracy of SVM and Adaboost using the instance subset are slightly lower than those of the whole training set, but the RBIS algorithm only uses 7.25% of instances to get good accuracy (i.e. 94.682% or 94.668%). This shows that the RBIS algorithm can reduce RR while maintaining accuracy. On the two balanced data sets, the accuracy of 1-NN using the instance subset is higher than that by the whole training set. This is because the instance subset is selected by the proposed instance selection algorithm and 1-NN. In addition to good ACC, the RR by the three classifiers and instance subsets are very small, which are 0.3% and 7.25%, respectively. This can prove that the RBIS algorithm can achieve a better balance between ACC and RR. On the other hand, from the perspective of average ACC, it is obvious that the average ACC by the instance subset is much higher than that by the whole training set on the DoS data set. Meanwhile, on the DDoS 2016 data set, the average ACC by the instance subset is only slightly higher than that obtained by the whole training set. This indicates that the RBIS algorithm can select optimal instances to improve ACC and reduce RR for IDS.

In Table 6, the experimental results demonstrate that the proposed RBIS algorithm is effective and can deal with balanced data problem. The RBIS algorithm is effective because it is based on the new instance representativeness, which is shown in Section 3.1. Through instance representativeness, the selected instances possess the information of the entire instances and are useful to improve ACC and reduce RR for IDS.

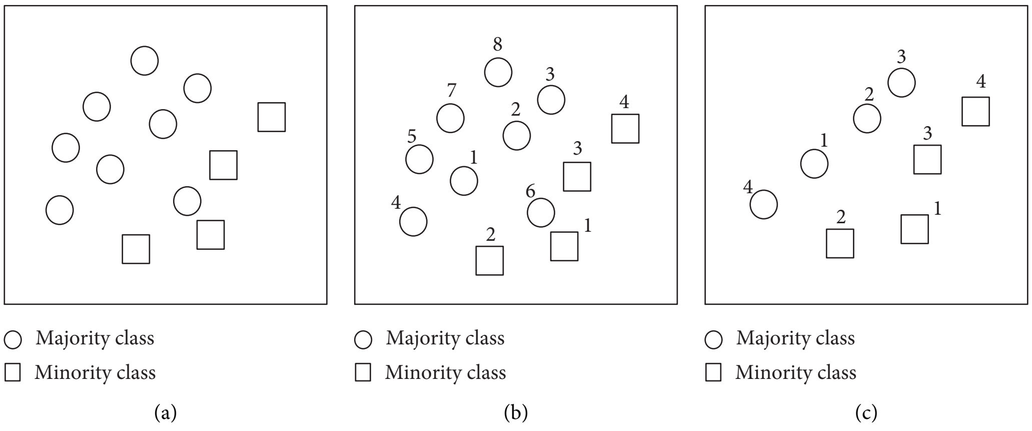


Figure 7: RBIS-IM selects the critical instances of all classes in the imbalanced data. (a) Original data. (b) Sort the instances according to their representativeness. (c) Select the same number of instances according to the parameter *t*.

Table 6: The efficiency of RBIS algorithm with 1-NN, SVM, and Adaboost is verified on the balanced data set.

Data set	The size of instances	Classifier	ACC (%)	RR	Average ACC (%)
DoS	20000	1-NN	61.000	100	63.692
		SVM	65.044		
		Adaboost	65.033		
	60	1-NN	99.913	0.3	93.362
		SVM	99.910		
		Adaboost	80.263		
DDoS 2016	20000	1-NN	90.823	100	93.653
		SVM	95.059		
		Adaboost	95.077		
	1450	1-NN	94.916	7.25	94.755
		SVM	94.682		
		Adaboost	94.668		

As shown in Table 7, on the imbalanced data sets, three common classifiers, which are 1-NN, SVM, and Adaboost, can obtain BA, RR, and average BA using the whole training set and the instance subset. Three imbalanced data sets are from the KDD Cup 99. On the Probe data set, using the instance subset, the three classifiers get good accuracy. Compared with the whole training set, the BA by the 1-NN classifier using instance subset is slightly lower, while BAby SVM and Adaboost are better. On U2R and R2L data sets, compared to using the whole training set BA of three common classifiers using instance subset is better. The experimental results prove that the RBIS-IM algorithm can achieve a better balance between BA and RR.

Besides, from the perspective of average BA, on the Probe data set, the average BA using the instance subset is slightly higher than that using the whole training set. On the U2R and R2L data sets, compared with the average BA using the whole training set, the average BA using the instance subset is greatly improved. Therefore, the experimental results on imbalanced data sets indicate that the RBIS-IM algorithm is effective and can obtain good RR while improving BA. This is because the RBIS-IM algorithm is also based on the new instance representativeness, which is shown in Section 3.1. Through instance representativeness, the optimal instances are selected to improve BA and reduce RR for IDS. And the experimental results display that the RBIS-IM algorithm can handle imbalanced data problem.

Table 7: The efficiency of RBIS-IM algorithm with 1-NN, SVM, and Adaboost is verified on the imbalanced data set.

Data set	The size of instances	Classifier	BA (%)	RR (%)	Average BA (%)
Probe	11550	1-NN	98.825	100	98.096
		SVM	99.104		
		Adaboost	96.359		
	2356	1-NN	97.887	20.398	98.148
		SVM	99.544		
		Adaboost	97.013		
U2R	10030	1-NN	49.970	100	50.079
		SVM	49.998		
		Adaboost	50.270		
	60	1-NN	61.580	0.598	61.632
		SVM	61.565		
		Adaboost	61.750		
R2L	11000	1-NN	80.465	100	74.860
		SVM	67.665		
		Adaboost	76.449		
	2000	1-NN	96.068	18.182	91.445
		SVM	87.859		
		Adaboost	90.407		

Table 8: Accuracy of ENN, ISAR, BNNT, CNNIR, RIS 1, and RBIS on the balanced data set.

Data set	ENN	ISAR	BNNT	CNNIR	RIS 1	RBIS
DoS	65.173	99.904	65.070	65.142	99.906	99.913
DDoS 2016	84.589	70.533	72.089	73.584	70.520	94.916
Mean	74.881	85.219	68.580	69.633	85.213	97.415

Table 9: Reduction rate of ENN, ISAR, BNNT, CNNIR, RIS 1, and RBIS on the balanced data set.

Data set	ENN	ISAR	BNNT	CNNIR	RIS 1	RBIS (%)
DoS	99.995	50.005	0.065	9.780	49.785	0.300
DDoS 2016	87.255	53.435	9.135	4.820	13.335	7.250
Mean	93.625	51.720	4.600	7.300	31.560	3.775

Tables 8 and 9 display the ACC and RR with the 6 instance selection algorithms on the balanced data sets. The proposed RBIS algorithm is compared with 5 algorithms: edited nearest neighbor (ENN) [22], ISAR [10], BNNT [8], CNNIR [9], and RIS 1 [11]. For ISAR and RIS 1, their instance selection algorithms are only used. On two balanced data sets, compared with the other 5 algorithms, the proposed RBIS algorithm achieves the best experimental results on ACC in Table 8. And the RBIS algorithm achieves the second RR on two balanced data sets in Table 9. In terms of average performance, it is obvious the RBIS algorithm achieves the best experimental results on ACC and RR. This indicates that the RBIS algorithm can achieve a better balance between ACC and RR. And, it can solve balanced data problem. Similarly, it proves that the RBIS algorithm is effective. In other words, the selected instances are optimal and contain the information of the whole instances. This is because four factors in the instance selection process are considered, which are shown in Section 3.1.

Table 10 shows the BA of 6 instance selection algorithms on the imbalanced data set. On the Probe data set, the BA of ENN, ISAR, RIS 1, and RBIS-IM algorithms are very close, and the biggest gap between them is less than 1%. This displays the RBIS-IM algorithm has the ability to distinguish between normal and attack instances. On the U2R and R2L data sets, the BA of the RBIS-IM algorithm is the best. Compared with other algorithms, the minimum gap is at least 10%. From the average BA, the average BA of the ENN, ISAR, and RIS 1 algorithms are very close, while the BA of the RBIS-IM algorithm is the best in Table 10. The experimental results prove that representative instances selected by RBIS-IM algorithm contain the information of the whole instances and the RBIS-IM algorithm can select representative instances to increase the BA for IDS. Moreover, the experimental results demonstrate that RBIS-IM algorithm can deal with imbalanced data problem.

Table 11 presents the RR of 6 instance selection algorithms on the imbalanced data set. On the Probe data set, the RR obtained by ISAR, CNNIR, and RIS 1 algorithms are very close. But, compared with ENN, other algorithms have a big gap with it. On the U2R data set, except for the ENN algorithm, the RR of other algorithms are very close and less than 1%. On R2L data, there is a small difference between the RR of the three algorithms, which are ISAR, CNNIR, and RIS 1 algorithms. From the average RR, the RR of the BNNT algorithm is the best. But, it is obvious that ENN gets poor RR (i.e. 99.879%). Since ENN is based on the nearest neighbor, ENN only removes instances near to the boundary and deletes limited instances of majority class. Moreover, ENN cannot deal with imbalanced data problem. The proposed RBIS-IM algorithm has good RR (i.e. 13.059%). This displays that the RBIS-IM algorithm can select small and representative instances to reduce RR. And the experimental results show that the RBIS-IM algorithm can deal with imbalanced data problem.

Table 10: BA of ENN, ISAR, BNNT, CNNIR, RIS 1, and RBIS-IM on the imbalanced data set.

Data set	ENN	ISAR	BNNT	CNNIR	RIS 1	RBIS-IM
Probe	98.789	98.059	70.510	87.175	98.059	97.887
U2R	49.980	49.592	50.755	50.000	49.642	61.580
R2L	80.434	79.956	53.797	63.592	85.238	96.068
Mean	76.401	75.869	58.354	66.922	77.646	85.178

Table 11: Reduction rate of ENN, ISAR, BNNT, CNNIR, RIS 1, and RBIS-IM on the imbalanced data set.

Data set	ENN (%)	ISAR (%)	BNNT (%)	CNNIR (%)	RIS 1 (%)	RBIS-IM (%)
Probe	99.896	13.680	0.537	10.312	13.680	20.398
U2R	99.950	0.489	0.578	0.680	0.160	0.598
R2L	99.791	9.455	0.945	6.327	9.000	18.182
Mean	99.879	7.875	0.687	5.773	7.613	13.059

The time complexity of 6 instance selection algorithms is present in Table 12. N represents the number of original instances. According to Table 12, the time complexity of the 6 algorithms is divided into two types. One is $O(N\log N)$, which are ENN, BNNT, and CNNIR algorithms. The other is $O(N^2)$, which are ISAR, RIS 1, RBIS, and RBIS-IM algorithms.

Figure 8 shows the relation of average ACC and average RR of 7 algorithms on the balanced data set and is based on Tables 6, 8, and 9. The 1-NN algorithm uses the whole training instances and the other 6 algorithms use the instance subset through their instance selection algorithms. On the balanced data set, the RBIS algorithm achieves the best in ACC and RR. Figure 8 suggests that the RBIS algorithm can select optimal instances to improve ACC and reduce RR for IDS. These optimal instances have the information for the entire instances.

Figure 9, which is based on Tables 7, 10, and 11, shows the relation of average BA and average RR of 7 algorithms on the imbalanced data set. It is obvious that RBIS-IM is the best on average BA. And Figure 9 suggests that the RBIS-IM algorithm can select optimal instances to increase BA and reduce RR for IDS. Although the average RR of the RBIS-IM algorithm is not the minimum, RBIS-IM algorithm can

Table 12: Time complexity of six algorithms.

ID	Algorithms	Time complexity
1	ENN	$O(N\log N)$
2	ISAR	$O(N^2)$
3	BNNT	$O(N\log N)$
4	CNNIR	$O(N\log N)$
5	RIS 1	$O(N^2)$
6	RBIS/RBIS-IM	$O(N^2)$

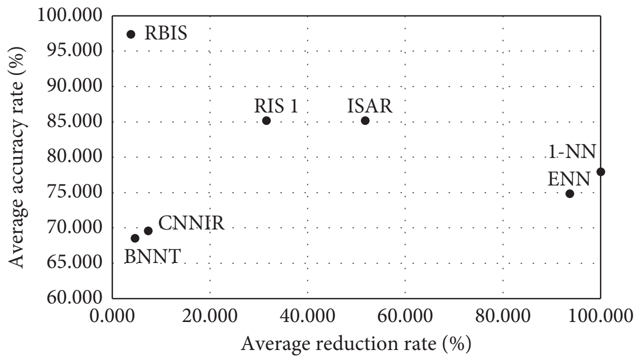


Figure 8: The relation of average ACC and average RR.

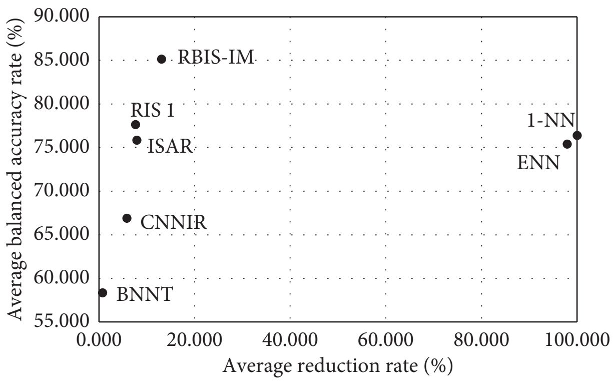


Figure 9: The relation of average BA and average RR.

achieve a good balance between average BA and average RR. Moreover, it is found that the RBIS-IM algorithm can handle imbalanced data problem.

5. Conclusions

In this paper, after analyzing the instance selection algorithm and its defects in intrusion detection, we propose a new representativeness of instance to determine the importance of an instance. Calculating the representativeness of instance, we consider not only the representativeness of instance in its category but also the representativeness of instances in different categories. These two representativenesses are equally important. Moreover, the influence of instances of different classes on selected instance is regarded as an advantage factor. To deal with balanced and imbalanced data problems, we propose the RBIS and RBIS-IM algorithms, respectively. In the process of instance selection, the proposed algorithms need not delete internal instances and noise instances. Compared with other algorithms on the benchmark data sets of intrusion detection, experimental results show that the two algorithms are effective. RBIS algorithm can achieve a better balance between accuracy (ACC) and reduction rate (RR). Similarly, the RBIS-IM algorithm can achieve a better balance between balanced accuracy (BA) and reduction rate (RR). Furthermore, it is also verified that the proposed representativeness of instance is correct and effective.

In future work, we intend to study how to automatically obtain the appropriate parameter t of the proposed approaches, which will reduce the training time of the algorithms. Moreover, obtaining the parameter t automatically can improve and enhance the effectiveness and applicability of the algorithms.

Data Availability

In this paper, two data sets are used for intrusion detection. They are public, which are the Knowledge Discovery and Data Mining (KDD) Cup 1999 data set and DDoS 2016 data set. The corresponding URLs are, respectively, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html and https://www.researchgate.net/publication/292967044_Dataset_Detecting_Distributed_Denial_of_Service_Attacks_Using_Data_Mining_Techniques.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grant No. 2017YFB0802300, in part by the Major Scientific and Technological Special Project of Guizhou Province under Grant No. 20183001, in part by the Foundation of Guizhou Provincial Key Laboratory of Public Big Data under Grant Nos. 2018BDKFJJ008 and 2018BDKFJJ020, and in part by the National Statistical Scientific Research Project of China under Grant Nos. 2018LY61 and 2019LY82.

References

[1] T. Phuoc, P. Tsai, T. Jan, and X. Kong, "Network intrusion detection using machine learning techniques," in *Proceedings of the 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, pp. 1–7, Vellore, India, February 2020.

[2] H. Hindy, D. Brosset, E. Bayne et al., "A taxonomy of network threats and the effect of current datasets on intrusion detection systems," *IEEE Access*, vol. 8, pp. 104650–104675, 2020.

[3] O. Adeleke, "Intrusion detection: issues, problems and solutions," in *Proceedings of the 3rd International Conference on Information and Computer Technologies (ICICT)*, pp. 397–402, San Jose, CA, USA, March 2020.

[4] J. Nalepa and M. Kawulok, "Selecting training sets for support vector machines: a review," *Artificial Intelligence Review*, vol. 52, pp. 857–900, 2019.

[5] Z. H. Zhu, Z. Wang, D. D. Li, and W. L. Du, "NearCount: selecting critical instances based on the cited counts of nearest neighbors," *Knowledge-Based Systems*, vol. 190, 2020.

[6] A. D. Haro-Garcia, G. Cerruela-Garcia, and N. Garcia-Pedrajas, "Instance selection based on boosting for instance-based learners," *Pattern Recognition*, vol. 96, 2019.

[7] C. Guo, Y.-J. Zhou, Y. Ping, S.-S. Luo, Y.-P. Lai, and Z.-K. Zhang, "Efficient intrusion detection using representative instances," *Computers & Security*, vol. 39, pp. 255–267, 2013.

[8] J. Li and Y. Wang, "A new fast reduction technique based on binary nearest neighbor tree," *Neurocomputing*, vol. 149, pp. 1647–1657, 2015.

[9] L. Yang, Q. Zhu, J. Huang, Q. Wu, D. Cheng, and X. Hong, "Constraint nearest neighbor for instance reduction," *Soft Computing*, vol. 23, no. 24, pp. 13235–13245, 2019.

[10] C. D. S. Pereira and G. D. C. Cavalcanti, "Instance selection algorithm based on a ranking procedure," in *Proceedings of the 2011 International Joint Conference on Neural Networks*, pp. 2409–2416, San Jose, CA, USA, July 2011.

[11] G. D. C. Cavalcanti and R. J. O. Soares, "Ranking-based instance selection for pattern classification," *Expert Systems with Applications*, vol. 150, 2020.

[12] H. Hmida, S. B. Hamida, A. Borgi, and M. Rukoz, "Hierarchical data topology based selection for large scale learning," in *Proceedings of the 2016 International 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)*, pp. 1221–1226, Toulouse, France, July 2016.

[13] J. Hamidzadeh, N. Kashefi, and M. Moradi, "Combined weighted multi-objective optimizer for instance reduction in two-class imbalanced data problem," *Engineering Applications of Artificial Intelligence*, vol. 90, 2020.

[14] L. Li, K. Y. Zhao, R. Z. Sun et al., "Parameter-free extreme learning machine for imbalanced classification," *Neural Processing Letters*, vol. 52, no. 3, pp. 1927–1944, 2020.

[15] H. X. Guo, Y. J. Li, J. Shang et al., "Learning from class-imbalanced data: review of methods and applications," *Expert Systems with Applications*, vol. 73, pp. 220–239, 2017.

[16] C.-H. Chou, B.-H. Kuo, and F. Chang, "The generalized condensed nearest neighbor rule as A data reduction method," in *Proceedings of the 18th International Conference on Pattern Recognition*, vol. 2, pp. 556–559, Hong Kong, China, August 2006.

[17] H. A. Fayed and A. F. Atiya, "A novel template reduction approach for the k-nearest neighbor method," *IEEE Transactions on Neural Networks*, vol. 20, no. 5, pp. 890–896, 2009.

[18] J. Arturo Olvera-López, J. Ariel Carrasco-Ochoa, and J. Francisco Martínez-Trinidad, "A new fast prototype selection method based on clustering," *Pattern Analysis & Applications*, vol. 13, no. 2, pp. 131–141, 2010.

[19] A. A. Akinyelu and A. E. Ezugwu, "Nature inspired instance selection techniques for support vector machine speed optimization," *IEEE Access*, vol. 7, pp. 154581–154599, 2019.

[20] A. Akinyelu and A. O. Adewumi, "On the performance of cuckoo search and bat algorithms based instance selection techniques for SVM speed optimization with application to E-fraud detection," *KSII Transactions on Internet and Information Systems*, vol. 12, no. 3, pp. 1348–1375, 2018.

[21] C. E. Brodley, "Recursive automatic bias selection for classifier construction," *Machine Learning*, vol. 20, pp. 63–94, 1995.

[22] I. Tomek, "An experiment with the edited nearest-neighbor rule," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 6, pp. 448–452, 1976.

[23] N. Jankowski and M. Grochowski, "Comparison of instances selection algorithms I. Algorithms survey," *International Conference on Artificial Intelligence and Soft Computing*, vol. 10, pp. 937–942, 2004.

[24] Q. Y. Wang, X. Q. Ouyang, and J. C. Zhan, "A classification algorithm based on data clustering and data reduction for intrusion detection system over big data," *KSII Transactions on Internet and Information Systems*, vol. 13, pp. 3714–3732, 2019.

[25] P. Ghosh, A. Saha, and S. Phadikar, "Penalty-reward based instance selection method in cloud environment using the concept of nearest neighbor," *Procedia Computer Science*, vol. 89, pp. 82–89, 2016.

[26] L. Yang, Q. Zhu, J. Huang, and D. Cheng, "Adaptive edited natural neighbor algorithm," *Neurocomputing*, vol. 230, pp. 427–433, 2017.

[27] N. García-Pedrajas, J. A. Romero del Castillo, and D. Ortiz-Boyer, "A cooperative coevolutionary algorithm for instance selection for instance-based learning," *Machine Learning*, vol. 78, no. 3, pp. 381–420, 2010.

[28] J. Li, Q. Zhu, and Q. Wu, "A parameter-free hybrid instance selection algorithm based on local Sets with natural neighbors," *Applied Intelligence*, vol. 50, no. 5, pp. 1527–1541, 2020.

[29] B. Jia and Y. Liang, "Anti-D chain: a lightweight DDoS attack detection scheme based on heterogeneous ensemble learning in blockchain," *China Communications*, vol. 17, no. 9, pp. 11–24, 2020.

[30] C. Guo, Y. Ping, N. Liu, and S. S. Luo, "A two-level hybrid approach for intrusion detection," *Neurocomputing*, vol. 214, 2016.

[31] University of California Department of Information and Computer Science, *KDD Cup 99 Intrusion Detection Dataset Task Description*, University of California Department of Information and Computer Science, Berkeley, CA, USA, 1999, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[32] M. Alkasassbeh, G. Al-Naymat, B. A. Ahmad, and M. Almseidin, "Detecting distributed denial of service attacks using data mining techniques," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 1, 2016.

[33] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 2019.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 6693726, 6 pages
https://doi.org/10.1155/2021/6693726

Research Article

Fail-Stop Group Signature Scheme

Jonathan Jen-Rong Chen,[1] Yi-Yuan Chiang,[2] Wang-Hsin Hsu,[3] and Wen-Yen Lin[3]

[1]*Department of Information Management, Vanung University, Taoyuan 302, Taiwan*
[2]*Department of Computer Science and Information Engineering, Vanung University, Taoyuan 302, Taiwan*
[3]*Department of Information Management, National Taichung University of Science and Technology, Taichung 404, Taiwan*

Correspondence should be addressed to Wen-Yen Lin; qqnice@nutc.edu.tw

Received 17 December 2020; Revised 25 January 2021; Accepted 29 January 2021; Published 9 February 2021

Academic Editor: Xiaokang Zhou

In this study, a fail-stop group signature scheme (FSGSS) that combines the features of group and fail-stop signatures to enhance the security level of the original group signature is proposed. Assuming that FSGSS encounters an attack by a hacker armed with a supercomputer, this scheme can prove that the digital signature is forged. Based on the aforementioned objectives, this study proposes three lemmas and proves that they are indeed feasible. First, how does a recipient of a digitally signed document verify the authenticity of the signature? Second, when a digitally signed document is under dispute, how can the group's manager determine the identity of the original group member who signed the document, if necessary, for an investigation? Third, how can one prove that the signature is indeed forged following an external attack from a supercomputer? Following an attack, the signature could be proved to be forged without exposing the key. In addition, the ultimate goal of the group fail-stop signature scheme is to stop using the same key immediately after the discovery of a forgery attack; this would prevent the attack from being repeated.

1. Introduction

Electronic documents are being increasingly used instead of paper to conduct official government and private business. Among other advantages, this benefits the environment by reducing the amount of paper being used. However, the use of electronic documents also increases the importance of using digital signatures to guarantee the validity, authenticity, and integrity of electronic documents and reduce the risk of documents being forged.

To cope with the wide range of potential uses for digital signature technology, the concept of group signing was proposed. A real-life example is considered to illustrate the process of using a group fail-stop signature scheme. The chief of Taiwan's Environmental Protection Administration, along with 19 other staff members of the agency, is eligible to digitally sign documents; these staff members include those accusing a subordinate unit of breaking the law. To safeguard the agency members' neutrality and protect them from interference, each staffer is required to activate a digital signature key when they release a statement or document representing the administration. The recipient of the document would be able to verify the authenticity of the digital signature. However, in the event that someone impeaches the integrity or validity of a digitally signed document, the identity of the individual who originally signed the document would remain a secret.

Companies or other entities cited for violations by the Environmental Protection Administration could file a complaint with the agency to deny that they had violated the law. As part of the review process, it might be necessary to determine the identity of the official who signed the original document making the accusation. Under the present scheme, only the manager in the group would have the ability to identify the person who signed the document. The manager, however, cannot pretend to be a member of another group to forge the digital signature.

Chaum and Van Heyst [1] concluded that there are three properties of group signatures. (i) Only members of the group can sign messages. (ii) The recipient can verify that it is a valid group signature but cannot discover the group member who signed the message. (iii) If necessary, the signature can be "opened," to reveal the person who signed the message.

The group signature scheme also has some favorable features that make it applicable in a range of fields. A digital signature can ensure the validity and authenticity of electronic documents. If the possibility of a document being forged could be reduced, or even if it were possible to prove that the digital signature was forged, the security level of the digital signature could then be enhanced. Another type of fail-stop signature scheme (FSS) can satisfy the aforementioned requirements.

Kitajima et al. [2] showed that an FSS has to have at least two security properties. (i) A scheme based on information-theoretic security has to be secure, even against a computationally unbounded adversary. (ii) If the computational assumption is broken, an honest signer should be able to prove that a signature is a forgery by virtue of information-theoretic security.

In this work, a fail-stop group signature scheme (FSGSS) is proposed. FSGSS combines all the functions and features of two schemes: group signature (GS) and FSS. This algorithm integrates the features of the two types of digital signatures, which strengthens its security level under the GS system. The combination scheme ensures that the group members can prove that a digital signature is indeed a forgery after supercomputer forgery attacks.

The remainder of this paper is organized as follows: Section 2 describes studies related to the present work. Section 3 presents our scheme, and Section 4 provides an analysis of the scheme and a discussion thereof. Finally, Section 5 concludes the paper and provides directions for future research.

2. Related Work

Desmedt proposed a group-oriented cryptosystem concept in 1987. In his dissertation [3], he noted that, in addition to entities that exist as individuals, there are entities comprising groups of several individuals, such as hospitals, schools, public institutions, and private companies. When these entities issue signed electronic documents, such as certificates, the concept of a digital signature becomes a mechanism to replace signatures on paper documents. Digital signatures could be placed on electronic diplomas, electronic medical records, and other official documents released by governmental agencies. The documents that carry digital signatures must have the following features: certainty of identity, nonrepudiation, and unforgeability.

Therefore, the design of the way keys are exchanged, the parameters of the exchanges become particularly important. Although each member in a group has a secret key, the group password must be reused. In other words, individuals in the group cannot exchange their keys during an operation. Instead, they exchange secondary keys derived from their main keys. This ensures the security of the main keys. In addition, members cannot export the group's master key. This ensures that this key remains secure. Chen and Yuanchi [4] developed a new and fast anonymous digital signature system by linking the LUC function with the complexities of discrete logarithms and factorization.

Conversely, multiple studies have focused on the security of conventional digital signature schemes that rely on a computational assumption. FSSS provide security for a sender against a forger with unlimited computational power by enabling the sender to provide a proof of forgery if it occurs. FSSs have been proposed in [5–10]. Chain [11] proposed that a fail-stop scheme could assert a victim's innocence, without exposing the $n = p \times q$ secret, and would guard against malicious behavior. More recently, Kitajima et al. [2] proposed a framework for FSS operating in a multisigner setting and called for a primitive fail-stop multisignature scheme. In other words, they combined threshold and fail-stop signatures. After the first aggregate signature scheme was proposed, several researchers attempted to propose more efficient versions of FSS by combining various schemes.

Recently, blockchain technology was used to realize the calculation and verification of the original GS algorithm. The calculation of the group certificate and signature recognition should be completed by the corresponding smart contract. This reduces the possibility of a joint attack. The newly added signature node no longer needs the approval of the center and only requires the approval of the majority node, realizing the true decentralization of signatures [12]. However, the decentralized GS scheme, based on blockchain, requires more calculation and is more expensive to implement smart contracts and applications without a decentralized network. Conversely, the blockchain-based smart contract is visible to all blockchain users. This leads to a situation where bugs, including security holes, are visible to all, yet may not be quickly fixed [13–15]. In particular, issues in Ethereum smart contracts include ambiguities and easy-but-insecure constructs in its contract language solidity, compiler bugs, Ethereum virtual machine bugs, attacks on the blockchain network, and the immutability of bugs; moreover, there is no central source documenting known vulnerabilities, attacks, and problematic constructs [14].

3. Proposed Scheme

3.1. Initialization. The system center (SC) chooses a primitive element g over the Galois field p_0, satisfying the following equation:

$$p_0 = 4p_1q_1 + 1, \tag{1}$$

where p_1, q_1 are large primitive. Let

$$n = p_1q_1. \tag{2}$$

Then, SC chooses a number $g_2 \in Z_n^*$, satisfying

$$g_2^{p_1} \equiv 1 (\bmod p_0), \tag{3}$$

where $\{g_2, p_0, n\}$ and $\{p_1, q_1\}$ are the public key and secret key of the SC, respectively. The details of the initialization process are shown in 0 (Figure 1).

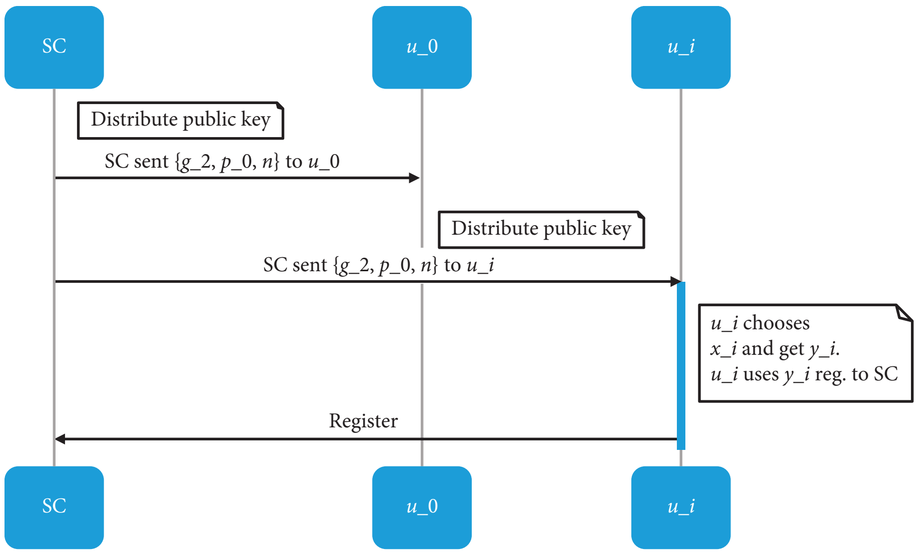

SC
u_0
u_i
Distribute public key
SC sent {g_2, p_0, n} to u_0
Distribute public key
SC sent {g_2, p_0, n} to u_i
u_i chooses x_i and get y_i. u_i uses y_i reg. to SC
Register
SC
u_0
u_i

Figure 1: Initialization of GFSS.

3.2. Group and Its Members. Without loss of generality, we assume a group and its members $u_i, 0 \le i \le l$, where u_0 is the manager of a group. The member registers to SC individually as follows:

u_i chooses a number x_i and calculates

$$g_2^{x_i} \equiv y_i (\bmod p_0), \tag{4}$$

u_i uses the y_i to register.

3.3. Parameters Exchange. Example 1: $u_i, 0 \le i \le l$, requests a part of the parameter from u_0; and then, u_0 chooses a number k and calculates

$$g_2^k \equiv r_1 (\bmod p_0), \tag{5}$$

$$u_0 \longrightarrow u_i\colon r_1. \tag{6}$$

This means that u_0 sends r_1 to u_i. u_i chooses a number b' and calculates

$$g_2^{b'} \equiv b (\bmod p_0), \tag{7}$$

$$r_1^b \equiv r_3 (\bmod p_0), \tag{8}$$

$$r_2 \equiv \frac{r_3}{b} (\bmod n), \tag{9}$$

$$u_i \longrightarrow u_o\colon r_2, \tag{10}$$

u_0 chooses a number a, satisfying the following equation:

$$a \equiv x_0 r_2 + ks (\bmod n), \tag{11}$$

$$u_0 \longrightarrow u_i\colon a, s. \tag{12}$$

After the aforementioned procedure is performed, if manager u_0 knows the parameters, k, a, x_0, and r_2, then s is known. It is to be noted that y_0, r_1 is the public key of u_0, where $g_2^{x_0} \equiv y_0 (\bmod p_0)$, based on equation (4). The detailed process of GFSS is shown in 0 (three-way handshake for exchange parameters) (Figure 2).

3.4. Signing Message m. Multiplying both sides of equation (11) with b, we obtain

$$ba \equiv x_0 (br_2) + (kb) s (\bmod n). \tag{13}$$

Multiplying both sides of equation (9) with b, we obtain

$$r_3 \equiv br_2 (\bmod n). \tag{14}$$

Using equations (13) and (14), we obtain

$$ba \equiv x_0 r_3 + (kb) s (\bmod n). \tag{15}$$

Then, we choose two numbers c, e and calculate

$$g_2^c \equiv r_5 (\bmod p_0), \tag{16}$$

$$g_2^e \equiv E (\bmod p_0). \tag{17}$$

Let

$$r_4 \equiv r_3 r_5 (\bmod p_0), \tag{18}$$

and

$$s_1 \equiv r_5 s (\bmod n). \tag{19}$$

Adding cs on both sides of equation (15), we acquire

$$ba + cs \equiv x_0 r_3 + (kb) s + cs \equiv x_0 r_3 + (kb + c) s (\bmod n). \tag{20}$$

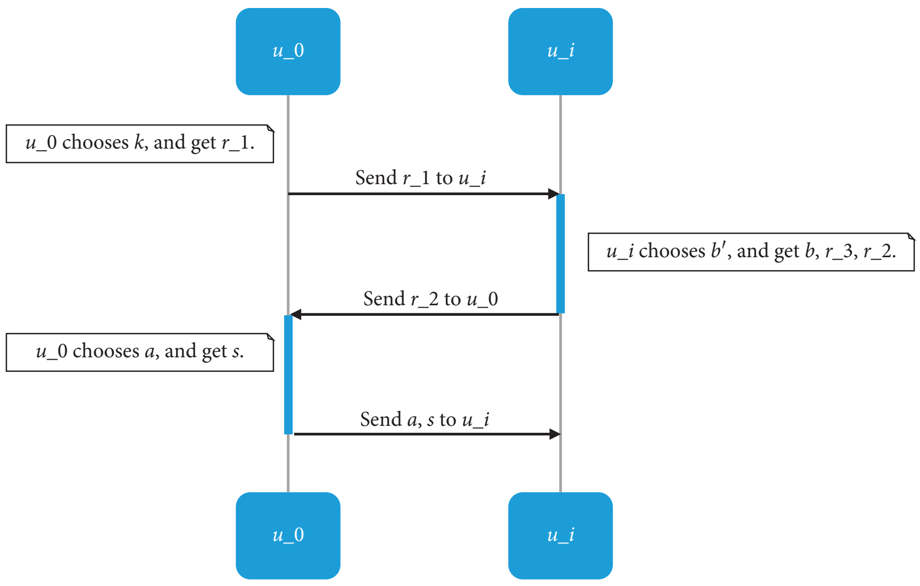

u_0
u_i
u_0 chooses k, and get r_1.
Send r_1 to u_i
u_i chooses b′, and get b, r_3, r_2.
Send r_2 to u_0
u_0 chooses a, and get s.
Send a, s to u_i
u_0
u_i

Figure 2: Three-way handshake for parameter exchange.

Using r_5 from equations (18) and (19) to multiply both sides of equation (20), we obtain

$$\begin{aligned}(ba+cs)r_5 &\equiv x_0\left(\frac{r_4}{r_5}\right)r_5+(kb+c)\left(\frac{s_1}{r_5}\right)r_5 \\ &\equiv x_0r_4+(kb+c)s_1\,(\mathrm{mod}\,n).\end{aligned} \quad (21)$$

Let

$$r_6=(ba+cs)r_5\,(\mathrm{mod}\,n), \quad (22)$$

and

$$m+r_6=cE+es_2\,(\mathrm{mod}\,n). \quad (23)$$

Assuming that the recipient of the message is R, u_i sends messages to R. It is to be noted that

$$u_i \longrightarrow R\colon \{m,c,E,r_4,r_6,s_1,s_2\}, \quad (24)$$

where R is used in the equations as follows:

$$g_2^{r_6}\equiv y_0^{r_4}r_4^{s_1}\,(\mathrm{mod}\,n), \quad (25)$$

$$g_2^{m+r_6}=g_2^{cE}E^{s_2}\,(\mathrm{mod}\,p_0). \quad (26)$$

The receiver accepts this digital signature if both equations (25) and (26) are valid. Otherwise, this digital signature is denied (Table 1).

4. Analysis and Discussion

In this section, we first introduce Lemma 1 to check the validity of a digital signature. Lemma 2 verifies whether a digital signature is activated by a group member. Lemma 3 shows that the attack method, mentioned by Susilo [7], will not succeed. There are several parameters after these procedures. We created a list of members holding parameters, as shown in 0. In this scheme, the members share partial parameters and maintain a few parameter(s). For example, manager u_0 holds only parameter k; member u_i only holds parameter b. In this case, someone creates a digital signature of u_0 and passes the verification; however, she/he is unaware of parameter b. That is, this person is a forger.

Lemma 1. *If u_0 and u_i are trusted authorities, then both equations (24) and (25) are valid.*

Proof. Using equation (22), we have

$$g_2^{r_6}\equiv g_2^{(ba+cs)r_5}\equiv g_2^{x_0r_4+(kb+c)s_1}\equiv g_2^{x_0r_4}g_2^{(kb+c)s_1}\,(\mathrm{mod}\,n). \quad (27)$$

There are two parts of the last term of the aforementioned equations; considering the first part and using equation (4), we have

$$g_2^{x_0r_4}\equiv\left(g_2^{x_0}\right)^{r_4}\equiv y_0^{r_4}\,(\mathrm{mod}\,n). \quad (28)$$

Considering the second part,

$$\begin{aligned}g_2^{(kb+c)s_1}&\equiv g_2^{(kb)s_1}g_2^{cs_1}\equiv\left(\left(g_2^k\right)^b\right)^{s_1}g_2^{cs_1}\equiv\left(\left(r_1\right)^b\right)^{s_1}g_2^{cs_1}\text{ via equation (5)}\\ &\equiv r_1^{bs_1}\left(g_2^c\right)^{s_1}\equiv r_1^{bs_1}r_5^{s_1}\text{ via equation (17)}\\ &\equiv r_3^{s_1}r_5^{s_1}\equiv\left(r_3r_5\right)^{s_1}\text{ via equation (8)}\\ &\equiv r_4^{s_1}\,(\mathrm{mod}\,n)\text{ via equation (18).}\end{aligned} \quad (29)$$

By combining equations (28) and (29), we obtain

$$g_2^{r_6}\equiv y_0^{r_4}r_4^{s_1}\,(\mathrm{mod}\,n). \quad (30)$$

Hence,

$$g_2^{m+r_6}=g_2^{cE}E^{s_2}\,(\mathrm{mod}\,p_0). \quad (31)$$

Therefore, both equations (25) and (26) are valid.

Table 1: List of members holding parameters.

	SC	u_o	u_i	R		SC	u_o	u_i	R
g_2	v	v	v	v	s		v	v	
p_o	v	v	v	v	m			v	v
n	v	v	v	v	c			v	v
y_i	v	v	v		E			v	v
r_1		v	v		r_4			v	v
b'			v		r_6			v	v
b			v		s_1			v	v
r_2		v	v		s_2			v	v
a		v	v						

Certain parameters are required to check whether message m has been sent by u_i. Hence, we obtain the following lemma: □

Lemma 2. *If u_0 and u_i are trusted authorities, then it implies that message m was sent from u_i by equation (8).*

Proof: The following should be noted:

(a) $r_1^b \equiv r_3 (\bmod p_0)$ from equation (8)
(b) $u_i \longrightarrow R$: $\{m, c, E, r_4, r_6, s_1, s_2\}$ from equation (24)
(c) $u_0 \longrightarrow u_i$: a, s from equation (12)
(d) $r_4 \equiv r_3 r_5 (\bmod p_0)$ from equation (18)
(e) $ba \equiv x_0 r_3 + (kb)s (\bmod n)$ from equation (15)
(f) $g_2^k \equiv r_1 (\bmod p_0)$ from equation (5)
(g) $g_2^{r_6} \equiv y_0^{r_4} r_4^{s_1} (\bmod n)$ from equation (25)
(h) $g_2^{m+r_6} = g_2^{cE} E^{s_2} (\bmod p_0)$ from equation (26)

Considering equation (19), $s_1 \equiv r_5 s$ (mod$n$), s_1 and s can be determined because of equations (12) and (24). Hence, we can obtain r_5 and r_3 via equations (18) and (24). Finally, b must be calculated (only u_i knows this parameter) because u_0 is unaware of b.

Considering equation (15), a, x_0, r_3, k, and s are known. It is not easy for anyone to obtain b; only the manager of group u_0 knows this value. In fact, it is a discrete logarithm problem when someone knows r_1, r_3 only by equation (8).

We conclude that u_0 can obtain b because u_0 already knows parts of parameters from u_i and has their own parameter k. Therefore, after checking equations (25) and (26), we can say that the message is sent by u_i. □

Lemma 3. *An attacker intercepting the message passed by the digital signature to adapt the method of Susilo et al. will not succeed.*

Proof: The following is to be noted:

(a) $u_i \longrightarrow R$: $\{m, c, E, r_4, r_6, s_1, s_2\}$ from equation (24)
(b) $g_2^{r_6} \equiv y_0^{r_4} r_4^{s_1} (\bmod n)$ from equation (25)
(c) $g_2^{m+r_6} = g_2^{cE} E^{s_2} (\bmod p_0)$ from equation (26)

If an attacker A intercepts the message as shown in equation (24) because A is unaware of parameters x_0, r_4, we assume that

$$\begin{aligned} g_2^{x_0'} &\equiv y_0 (\bmod p_0), \\ g_2^{r_4'} &\equiv r_4 (\bmod p_0). \end{aligned} \tag{32}$$

Attacker A can easily forge m^* for suitable parameters $\{c', E', s_2'\}$ such that both equations (25) and (26) are valid. In other words, the digital signature passes the test of Lemma 2. After the procedure of Lemma 2 is performed, a nontrivial factor of n can be found by computing $GC\ D(b, b^*, n)$. We note that the probability of b being equal to b^* is $(1/q_0)$. Therefore, it is proved that m^* is not sent by the group members. □

5. Conclusions and Future Work

In this study, we propose a novel FSGSS. This algorithm integrates the features of two types of digital signature, which strengthens its security level under the GS system. The proposed FSGSS ensures that the group members can prove that a digital signature is indeed a forgery after supercomputer forgery attacks. In addition to discussing the integration of these two digital signatures, this dissertation highlights three proposed Lemmas and proves that they are feasible. Lemma 1 verifies an FSGSS digital signature. Lemma 2 is used by the group manager, when needed, to determine the identity of the group member who originally created the digital signature. Finally, this dissertation proposes Lemma 3. When the digital signature is found to be forged, members of the group can prove this fact.

The ultimate goal of the group fail-stop signature scheme is to stop using the same key immediately after the discovery of a forgery attack; this would prevent the attack from being repeated. That is, the "key" considered in this study is parameter b used by u_i. If the parameters need to be changed each time an entity is under attack, the process of replacing the parameters is equivalent to reexecuting the exchange parameter program. Therefore, in future work, we plan to design a scheme wherein we need not directly expose key b; we can then prove that a certain number of signatures are forged, which will enhance the efficiency of GFSS.

Data Availability

No data were used to support the findings of this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

Wang-Hsin Hsu passed away in March 15, 2020. We would like to express our gratitude to him for his contribution to this paper. We will always miss you and continue your unfinished wishes. Hope you rest in peace. This research is partially supported by the “Higher Education Sprout Project,” Ministry of Education, Taiwan.

References

[1] D. Chaum and E. Van Heyst, “Group signatures,” in *Workshop On the Theory And Application Of Cryptographic Techniques*, Springer, Berlin, Germany, 1991.

[2] N. Kitajima, N. Yanai, T. Nishide, G. Hanaoka, and E. Okamoto, “Constructions of fail-stop signatures for multi-signer setting,” in *Proceedings of the 2015 10th Asia Joint Conference On Information Security*, IEEE, Kaohsiung City, Taiwan, May 2015.

[3] Y. Desmedt, “Society and group oriented cryptography: a new concept,” in *Proceedings of the Conference On the Theory And Application Of Cryptographic Techniques*, Springer, Amsterdam, The Netherlands, April 1987.

[4] J. J.-R. Chen and L. Yuanchi, “A traceable group signature scheme,” *Mathematical Computer Modelling*, vol. 31, no. 2-3, pp. 147–160, 2000.

[5] T. P. Pedersen and B. Pfitzmann, “Fail-stop signatures,” *SIAM Journal on Computing*, vol. 26, no. 2, pp. 291–330, 1997.

[6] K. Schmidt-Samoa, “Factorization-based fail-stop signatures revisited,” in *Proceedings of the International Conference On Information And Communications Security*, Springer, Malaga, Spain, October 2004.

[7] W. Susilo, “A new and efficient fail-stop signature scheme,” *The Computer Journal*, vol. 43, no. 5, pp. 430–437, 2000.

[8] E. Van Heyst and T. P. Pedersen, “How to make efficient fail-stop signatures,” in *Workshop On the Theory and Application of Cryptographic Techniques*, Springer, Berlin, Germany, 1992.

[9] X. Zhou, W. Liang, S. Shimizu, J. Ma, and Q. Jin, “Siamese neural network based few-shot learning for anomaly detection in industrial cyber-physical systems,” *IEEE Transactions on Industrial Informatics*, vol. 1, 2021.

[10] X. Zhou, Y. Hu, W. Liang, J. Ma, and Q. Jin, “Variational LSTM enhanced anomaly detection for industrial big data,” *IEEE Transactions on Industrial Informatics*, vol. 1, 2020.

[11] K. Chain, “An improved fail-stop signature scheme based on dual complexities,” *International Journal of Innovative Computing, Information and Control*, vol. 10, no. 2, pp. 535–544, 2014.

[12] Y. Cao, “Decentralized group signature scheme based on blockchain,” in *Proceedings of the 2019 International Conference On Communications, Information System And Computer Engineering (CISCE)*, Haikou, China, July 2019.

[13] M. E. Peck, “Ethereum’s $150-million blockchain-powered fund opens just as researchers call for a halt,” 2016, https://spectrum.ieee.org/tech-talk/computing/networks/ethereums-150-million-dollar-dao-opens-for-business-just-as-researchers-call-for-a-moratorium.

[14] N. Atzei, M. Bartoletti, and T. Cimoli, “A survey of attacks on ethereum smart contracts (sok),” in *Proceedings of the International Conference on Principles of Security and Trust*, Springer, Uppsala, Sweden, April 2017.

[15] X. Zhou, W. Liang, K. I.-K. Wang, H. Wang, L. T. Yang, and Q. Jin, “Deep-Learning-Enhanced human activity recognition for internet of healthcare things,” *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6429–6438, 2020.