Wireless Communications and Mobile Computing

Models, Technologies, and Applications of Intelligent Defense in Ubiquitous Network Environments

Lead Guest Editor: Yawen Chen
Guest Editors: Lei Zhang and Lingling Lv

Models, Technologies, and Applications of Intelligent Defense in Ubiquitous Network Environments

Wireless Communications and Mobile Computing

Models, Technologies, and Applications of Intelligent Defense in Ubiquitous Network Environments

Lead Guest Editor: Yawen Chen
Guest Editors: Lei Zhang and Lingling Lv

Contents

Hindawi
Wireless Communications and Mobile Computing
Volume 2023, Article ID 9785959, 1 page
https://doi.org/10.1155/2023/9785959

Retraction

Retracted: Intelligent Application of Raw Material Supply Chain Planning System Based on Genetic Algorithm

Wireless Communications and Mobile Computing

Received 17 October 2023; Accepted 17 October 2023; Published 18 October 2023

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope
(2) Discrepancies in the description of the research reported
(3) Discrepancies between the availability of data and the research described
(4) Inappropriate citations
(5) Incoherent, meaningless and/or irrelevant content included in the article
(6) Peer-review manipulation

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] F. Xu, Y. Zhang, Y. Su, J. Li, and J. Zhu, "Intelligent Application of Raw Material Supply Chain Planning System Based on Genetic Algorithm," *Wireless Communications and Mobile Computing*, vol. 2022, Article ID 5054529, 13 pages, 2022.

Hindawi
Wireless Communications and Mobile Computing
Volume 2023, Article ID 4386621, 11 pages
https://doi.org/10.1155/2023/4386621

Research Article

A Network Key Node Identification Method Based on Improved Multiattribute Fusion

Bo Chen,[1,2] Rui Tong,[1] Yufeng Chen,[1] Panling Jiang,[1] Xiue Gao,[3] and Hang Tao[1]

[1]*School of Electrical and Information Engineering, Hubei University of Automotive Technology, Shiyan 442002, China*
[2]*School of Electronic and Electrical Engineering, Lingnan Normal University, Zhanjiang 524048, China*
[3]*School of Computer Science and Intelligence Education, Lingnan Normal University, Zhanjiang 524048, China*

Correspondence should be addressed to Yufeng Chen; chenyf_dy@huat.edu.cn

Received 24 May 2022; Revised 18 August 2022; Accepted 3 October 2022; Published 2 May 2023

Academic Editor: Lei Zhang

Considering the shortcomings of the existing network key node identification methods based on multiattribute fusion, which have single evaluation methods and low decision accuracy, combined with the advantages of the high accuracy of TOPSIS (Technique for Order Preference by Similarity to an Ideal Solution) algorithm and the applicability of grey relational analysis method for incomplete information evaluation, the concept of relative closeness is proposed, and nodes are ranked in importance based on the relative closeness; a key node identification method algorithm based on improved multiattribute fusion is designed. First, the identification problem of key nodes is transformed into multiattribute decision-making method, and the decision matrix is obtained. Second, the weighting matrix is obtained by weighting them in both subjective and objective dimensions, the relative closeness is calculated for the weighting matrix. Finally, sort the network nodes by relative closeness, and network performance simulation experiments are designed using various combinations of evaluation methods and key node identification methods. The simulation results show that this method is more adaptable and improves the identification accuracy of the network key nodes.

1. Introduction

Complex networks have attracted much attention from researchers owing to their scale-free, fragility, self-organization, and other characteristics. In real life, many systems, such as social networks, power networks, and transportation networks, can be represented using complex networks. The key nodes of various types of complex networks play an important role in the network structure and function, and knowing how to identify the key nodes is crucial for complex network reliability. Mining important nodes in social networks can help with decision-making in areas such as public opinion monitoring and advertising and marketing [1, 2]. In transportation networks, identifying key nodes of transportation hubs in advance can effectively prevent traffic congestion problems [3, 4]. In power networks, identifying key nodes of power networks in advance can implement protection and maintenance measures for key grid nodes [5]; as can be seen, discovering key nodes of various types of complex networks has high practical value.

Many network critical node identification methods have been proposed, such as degree centrality, betweenness centrality, K-shell, and structural holes. On this basis, many scholars have proposed improvements to these identification methods. Considering the characteristics of directed weighted networks, Zhao et al. [6] proposed JP-degree centrality in view of the shortage of traditional degree centrality that cannot be applied to directed weighted networks. The literature addressed the lack of degree centrality, which is difficult to directly apply to community networks, and proposed semilocal centrality algorithm that combined community structure with node degree [7]. Wang et al. [8] considered the form of vectors and proposed multiorder neighbor shell vector centrality. Wang et al. [9] introduced hierarchical flow betweenness to improve the structural hole method. Wang et al. [10] proposed and applied an improved efficiency centrality method to weighted networks. Hu et al. [11] proposed an importance identification method for network nodes based on neighborhood information entropy. Considering that the PageRank algorithm is only suitable for static networks, Xu and Wang [12] proposed an ALR algorithm that combines H-index and LeaderRank to adapt to changes in network topology. The aforementioned methods are single-index identification and their improvement, which describe the importance of nodes in a network from different perspectives; however, different networks have different structural characteristics, and even different parts of the same network have different structures. Therefore, a single metric identification and improving its performance in different networks can be difficult. For this reason, many scholars have applied multiattribute decision-making to key node identification.

The multiattribute decision method uses multiple key node identification methods as metrics for comprehensive evaluation of nodes, which no longer emphasizes the influence of a single factor one-sidedly and is often used for comprehensive evaluation of network key nodes. Yu et al. [13] and Liu et al. [14] used the subjective weighting method to determine the weights of evaluation metrics and applied it to network key node identification in combination with TOPSIS (Technique for Order Preference by Similarity to an Ideal Solution) method to achieve better results; Yang et al. [15] used information on the decision matrix to objectively assign weights to each metric. Other studies introduced the SIR (Susceptible Infected Recovered) model to dynamically calculate the weights of each evaluation metric [16, 17]. A combined centrality to the gravitational law to comprehensively identify the influence of the network nodes was applied [18]. Some studies proposed further improvements to the multiattribute decision method in terms of weights [19, 20]. The above methods often use only one method, TOPSIS or Vikor, when calculating the evaluation results, and directly use the sample data for analysis; however, in real networks, it is difficult to ensure the integrity of information acquisition in the system, and the decision results will cause certain errors.

The grey relational analysis is a method for measuring the degree of association between factors based on the similarity or dissimilarity of their trends; that is, “grey relational degree” is a simple and reliable method in the analysis system that can solve this problem well. It is better suited to situations where the system information is incomplete. Therefore, this study proposes a key node identification method based on improved multiattribute fusion, which fully combines the advantages of the high accuracy of the TOPSIS algorithm and the grey relational analysis method for incomplete information evaluation and improves the identification accuracy of the network key nodes.

The main contributions of this study are in the following areas:

(1) An improved multiattribute fusion key node identification method combining TOPSIS and grey relational analysis is proposed

(2) The subjective and objective comprehensive weighting method is proposed, and the relative closeness is proposed to calculate the evaluation results

(3) Example algorithms were designed to analyze and compare the network performance of different combinations of node importance evaluations and different networks

This study is organized as follows: Section 2 introduces several typical key node identification methods. Section 3 elaborates the algorithm flow and specific steps of this study. Section 4 illustrates the effectiveness and applicability of the proposed method by designing different simulation experiments. Section 5 is the conclusion section, which summarizes the research and provides future directions.

2. Metrics for Evaluating the Importance

The typical key node identification method is used as an evaluation metric of node importance. The node importance metrics are as follows.

2.1. Degree Centrality (DC).

2.1. Degree Centrality (DC). Degree centrality is the most direct metric to characterize the centrality of a node in network analysis. The larger the node degree of a node, the higher the DC of the node, and the more important the node is in the network. The formula is as follows:

$$\mathrm{DC}_i = \frac{k_i}{N-1}, \tag{1}$$

where N is the number of nodes and k_i is the degree of the node i.

2.2. Structural Hole (SH). There is no direct or indirect connection between the two nodes in the network, so the vacancy between the nodes is a structural hole. Burt proposed to calculate the network constraint coefficient to measure the structural hole, and the formula is as follows:

$$\mathrm{SH}_i = \sum_j \left(P_{ij} + \sum_{q \neq i \neq j} P_{iq} P_{qj} \right)^2, \tag{2}$$

where P_{ij} is the ratio of the energy invested by node i to maintain the neighbor relationship with node j to the total energy and q is the indirect node between node i and node j. The smaller the constraint coefficient SH_i, the larger the SH and the more important the position of the node.

2.3. Closeness Centrality (CC). Closeness centrality reflects the proximity between a node and other nodes in the network. The formula is as follows:

$$\mathrm{CC}_i = \frac{N}{\sum_{j=1}^{N} d_{ij}}, \tag{3}$$

where N is the number of nodes and d_{ij} is the shortest distance between node i and node j. The higher the value of the CC of a node, the more important its position is.

2.4. Betweenness Centrality (BC). Betweenness centrality is a measure of graph centrality based on the shortest path. The centrality of a node is the number of shortest paths through that node. The formula is as follows:

$$\mathrm{BC}_i = \sum_{j\neq i\neq k\in V}^{N} \frac{g_{jk}(i)}{g_{jk}}, \tag{4}$$

where $g_{jk}(i)$ represents the number of shortest paths between nodes j and k through node i and g_{jk} represents the number of all shortest paths between nodes j and k. The larger the value of BC, the more important position the node assumes in the information flow of the network.

The degree centrality is simple and suitable for all kinds of basic networks, but not very accurate. Since the structural hole can calculate the dependence of nodes on other nodes in the network, and the regular network has a high clustering coefficient, clustering coefficient is a measure of how well a node's neighbors are connected to each other. Structural holes are suitable for regular networks and small-world networks with a high degree of clustering coefficient. Degree centrality and structural hole only utilizes the local features of the network, and it has certain limitations. Closeness centrality and betweenness centrality make use of the global features of the network, that is, the position of a node in the whole structure. The closeness centrality can avoid being affected by the distance extremes generated by individual isolated nodes. The betweenness centrality represents the degree of independence between nodes; they are suitable for random networks and scale-free networks. Most of the real networks cover all or part of the characteristics of the above-mentioned standard networks. For example, scale-free networks are universal, and social networks, biological networks, trade networks, and other types of networks have scale-free network characteristics. Therefore, this study integrate the locality and globality of several metrics, combines the advantages of each method, and applies these metrics to multi-indicator fusion.

3. The Specific Flowchart

The idea of the node importance identification method based on multiattribute decision-making is to regard the nodes in the complex network as a scheme, regard multiple basic evaluation metrics for evaluating the importance of nodes as attributes of each scheme, and then judge the importance of nodes through the decision results. The specific implementation method of the method is as follows.

3.1. Constructing the Decision Matrix. Let there be N nodes in the complex network, and then, the corresponding set of decision schemas can be denoted as $A = \{A_1, A_2, \cdots A_N\}$. If there are m metrics to evaluate the importance of each node, the corresponding set of schema attributes is denoted as $S = \{S_1, S_2, \cdots S_m\}$. The value of the jth metric of the ith node is denoted as $A_i(S_j)$, which constitutes the decision matrix.

$$X = \begin{pmatrix} A_1(S_1) & \cdot & \cdot & A_1(S_m) \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ A_N(S_1) & \cdot & \cdot & A_N(S_m) \end{pmatrix}. \tag{5}$$

Table 1: Comparison matrix of node importance metrics.

	DC	SH	CC	BC
DC	1	0	0	0
SH	2	1	1	0
CC	2	1	1	0
BC	2	2	2	1

Then, the metrics were regularized as follows:

$$\begin{aligned} r_{ij} &= \frac{A_i(S_j)}{A_i(S_j)^{\max}}, \\ r_{ij} &= \frac{A_i(S_j)^{\min}}{A_i(S_j)}, \end{aligned} \tag{6}$$

where

$$\begin{aligned} A_i(S_j)^{\max} = \max\left\{A_i(S_j), (1 \le i \le N)\right\} A_i(S_j)^{\min} \\ = \min\left\{A_i(S_j), (1 \le i \le N)\right\}. \end{aligned} \tag{7}$$

The standardized decision matrix is denoted as $R = (r_{ij})_{N*m}$.

3.2. Calculation of the Weight of each Metric

3.2.1. The AHP Method to Calculate Subjective Weights of Metrics. First, the three-scale method is used to build a comparison matrix for each metric after a two-by-two comparison of each metric. Table 1 lists the values in the comparison matrix B constructed according to the three-scale method in the following equation:

$$B = (b_{ij}) = \begin{cases} 2, & \text{Metric } i \text{ is more important than metric } j, \\ 1, & \text{Metric } i \text{ is as important as metric } j, \\ 0, & \text{Metric } j \text{ is more important than metric } i. \end{cases} \tag{8}$$

The comparison matrix is then used to construct a judgment matrix C using the difference method, and a consistency test is performed. Finally, the metrics' weights are obtained by the following method.

$$\begin{aligned} M_i &= \prod_{j=1}^{4} c_{ij}, \\ W_i &= \sqrt[4]{M_i}, \end{aligned} \tag{9}$$

where $C = (c_{ij})$. After normalizing W_i, the final weight can be obtained.

3.2.2. The Entropy Method for Calculating Objective Weights of Metrics. In information theory, entropy is used to determine the degree of dispersion of a metric. The greater the degree of dispersion of a metric, the greater its weight in the composite weight. This is a classical method for assigning weights. The entropy of the jth metric is calculated as follows:

$$e_j = -\frac{1}{\ln |N|}\sum_{i=1}^{|N|} p_{ij} \ln \left(p_{ij}\right), \quad j = 1, 2, \cdots m,$$
$$p_{ij} = \frac{r_{ij}}{\sum_{i=1}^{N} r_{ij}}, \quad i = 1, 2, \cdots N, j = 1, 2, \cdots m, \tag{10}$$

where e_j is the entropy of the jth column of metrics and p_{ij} is the weight of the jth indicator of the ith node in that column of metrics. Finally, the weight of each is determined as follows:

$$w_j = \frac{1 - e_j}{\sum_{j=1}^{m}\left(1 - e_j\right)}, \tag{11}$$

where $1 - e_j$ is the information entropy redundancy and w_j satisfies $\sum w_j = 1$.

3.2.3. Calculation of Composite Weights. This algorithm integrates the subjective and objective weights and reasonably allocates the subjective and objective weight coefficients, and the final combined weight is expressed as follows:

$$w_j = \alpha w_{j1} + \beta w_{j2}, \tag{12}$$

where w_{j1} is the subjective weight calculated using the AHP method and w_{j2} is the objective weight calculated using the entropy weight method. α and β are the weighting coefficients, solved as follows:

$$\alpha = \frac{\sum_{i=1}^{N}\sum_{j=1}^{m} w_{j1} r_{ij}}{\sqrt{\left(\sum_{i=1}^{N}\sum_{j=1}^{m} w_{j1} r_{ij}\right)^2 + \left(\sum_{i=1}^{N}\sum_{j=1}^{m} w_{j2} r_{ij}\right)^2}},$$
$$\beta = \frac{\sum_{i=1}^{N}\sum_{j=1}^{m} w_{j2} r_{ij}}{\sqrt{\left(\sum_{i=1}^{N}\sum_{j=1}^{m} w_{j1} r_{ij}\right)^2 + \left(\sum_{i=1}^{N}\sum_{j=1}^{m} w_{j2} r_{ij}\right)^2}}. \tag{13}$$

The coefficients obtained from the above equation are then normalized to give the final weight coefficients α and β.

3.3. Relative Closeness Calculation. First, we construct the weighting matrix Y as follows:

$$Y = \left(y_{ij}\right) = \left(w_j r_{ij}\right) = \begin{pmatrix} w_1 r_{11} & \cdots & w_m r_{1m} \\ . & \cdots & . \\ . & \cdots & . \\ w_1 r_{N1} & \cdots & w_m r_{Nm} \end{pmatrix}. \tag{14}$$

3.3.1. Calculate the Euclidean Distance. The positive and negative ideal decision schemas are determined from the matrix Y. The Euclidean distance between each option and the positive and negative ideal options is then calculated using the following equation:

$$D_i^+ = \left[\sum_{j=1}^{m}\left(y_{ij} - y_j^{\max}\right)^2\right]^{1/2},$$
$$D_i^- = \left[\sum_{j=1}^{m}\left(y_{ij} - y_j^{\min}\right)^2\right]^{1/2}. \tag{15}$$

3.3.2. Calculate the Grey Relational Degree. Calculate the grey relational coefficient between the ith sample and the positive ideal sample on the jth metric based on the weighted normalization matrix:

$$S_{ij}^+ = \frac{\min_i \min_j \Delta y_{ij} + \rho \max_i \max_j \Delta y_{ij}}{\Delta y_{ij} + \rho \max_i \max_j \Delta y_{ij}}, \tag{16}$$

where $\Delta y_{ij} = |y_j^+ - y_{ij}|$, $\min_i \min_j \Delta y_{ij}$ is the two-level minimum difference, $\max_i \max_j \Delta y_{ij}$ is the two-level maximum difference, and $\rho \in [0, 1]$ is the discrimination coefficient; the smaller the discrimination coefficient, the greater the difference between the correlation coefficients and the stronger the discrimination ability, usually taken as 0.5. Then, the grey relational coefficient matrix of each sample and the positive ideal sample is determined as follows:

$$W^+ = \begin{pmatrix} s_{11}^+ & \cdots & s_{1m}^+ \\ . & \cdots & . \\ . & \cdots & . \\ s_{N1}^+ & \cdots & s_{Nm}^+ \end{pmatrix}. \tag{17}$$

The grey relational of the ith sample with the positive ideal sample is expressed as follows:

$$W_i^+ = \frac{1}{m}\sum_{j=1}^{m} s_{ij}^+. \tag{18}$$

Similarly, replacing $\Delta y_{ij} = |y_j^- - y_{ij}|$ in Equation (16) with Δy_{ij}, the grey relational degree of each sample with the negative ideal sample can be obtained.

$$W_i^- = \frac{1}{m}\sum_{j=1}^{m} s_{ij}^-. \tag{19}$$

3.3.3. Calculate the Relative Closeness. First, the Euclidean distance and grey relational degree are made dimensionless separately as follows:

$$\varphi_i = \frac{\Phi_i}{\max_{1\le i\le N}\left(\Phi_i\right)}, \tag{20}$$

where Φ_i represents the D_i^+, D_i^-, W_i^+, and W_i^- derived and is represented by d_i^+, d_i^-, w_i^+, and w_i^-, respectively, after it is made dimensionless. Combining the Euclidean distance

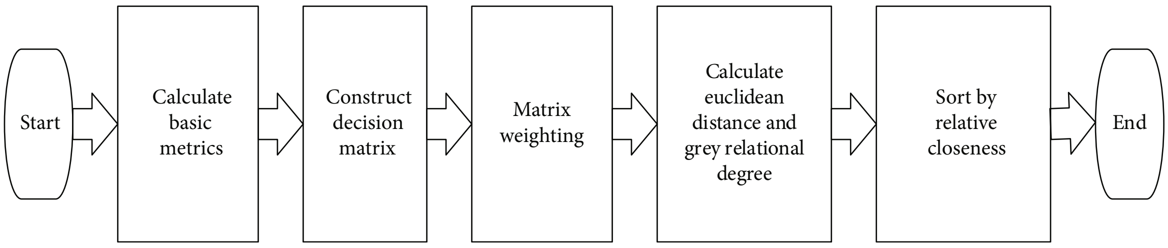

Start
Calculate basic metrics
Construct decision matrix
Matrix weighting
Calculate euclidean distance and grey relational degree
Sort by relative closeness
End

Figure 1: Flowchart of the algorithm.

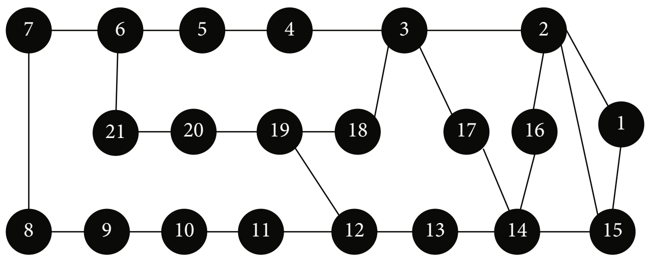


Figure 2: ARPA network topological structure.

and the grey relational degree, we obtain the following:

$$T_i^+ = e_1 d_i^- + e_2 w_i^+,$$
$$T_i^- = e_1 d_i^+ + e_2 w_i^-, \quad (21)$$

where $e_1 + e_2 = 1$. The values of e_1 and e_2 can be set according to preferences. This study takes $e_1 = e_2 = 0.5$. Finally, the relative closeness is calculated, and the final comprehensive evaluation result can be obtained using the following equation:

$$\delta_i = \frac{T_i^+}{T_i^+ + T_i^-}. \quad (22)$$

3.4. Steps of Algorithm. The steps of the key node identification algorithm based on improved multiattribute fusion are shown in Figure 1.

Step 1. Calculate basic evaluation metrics of the network, such DC and SH.

Step 2. Construct decision matrix and normalize it to form a multiattribute decision matrix.

Step 3. Substitute the weights of each metric obtained from the combination of subjective and objective methods into decision matrix to obtain the weighted matrix.

Step 4. Calculate Euclidean distance and grey relational degree by using weighted matrix.

Step 5. The relative closeness is calculated to get the comprehensive importance of the nodes, which is ranked from largest to smallest. The larger the closeness, the higher the importance of the node in the network.

Table 2: Ranking results of node importance evaluation on ARPA network.

Ranking	DC	SH	CC	BC	Proposed method
1	3 2 14	3	3	3	3
2	6 12 15 19	14	19	12	12
3	1 4 5 7 8 9 10 11 13 16 17 18 20 21	12 19	12	19	19
4		6	18	6	14
5		2	4 13 14	4	6
6		15	17	14	2
7		17	2 20	13	4
8		13 18	5 6	5	13
9		4	11	11	5
10		5 7 11	15	2	11

4. Algorithm Analysis

4.1. Evaluation Methodology. Different network models are deliberately attacked, and the key nodes identified by each algorithm are removed one by one. The impact of removing key nodes on the network is measured using three indexes: average network efficiency, network connectivity coefficient, and maximum-connected subgraph ratio, and then, the recognition accuracy of different algorithms is compared.

4.1.1. The Average Network Efficiency. This is defined as the average of the sum of the reciprocal of the distances between any two points in the network, which reflects the ability of information to flow in the network. The higher the average network efficiency, the better the integrity of the network when it is under attack. It is defined as follows:

$$\eta = \frac{1}{N \times (N-1)} \times \sum_{i \neq j} \frac{1}{d_{ij}}, \quad (23)$$

where d_{ij} denotes the distance between nodes i, j.

The accuracy of the algorithm's identification is determined using the average rate of decline in the network's efficiency. The faster the decline, the faster the network is down, and the more important the identified nodes are.

4.1.2. Network Connectivity Coefficient. This measures the relationship between the network invulnerability and the number of connected branches. It can analyze the partitioning

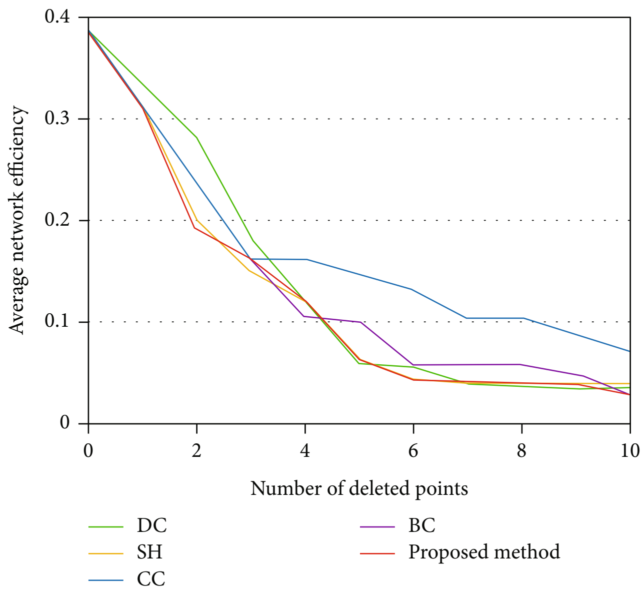


Figure 3: Comparison of average network efficiency of different methods.

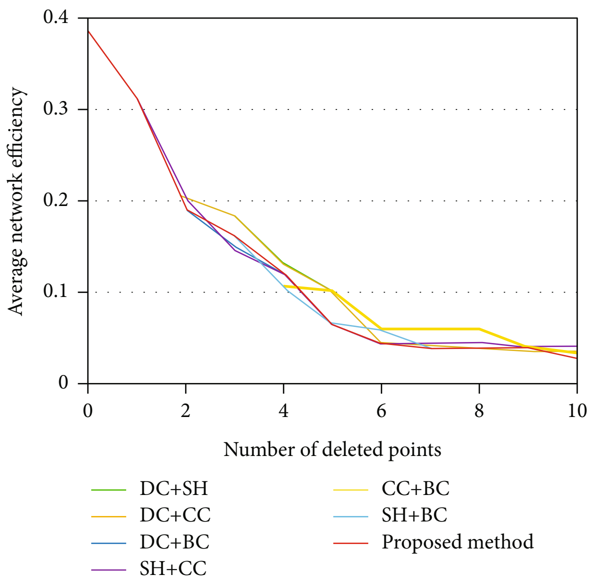


Figure 4: Comparison of different combinations of two metrics with the method in this study.

of the network after the deletion of nodes. The smaller the network connectivity coefficient, the more severely segmented the network is and the worse the invulnerability is, indicating that the deleted node is more important. The expression of network connectivity coefficient is expressed as follows:

$$\Phi = \frac{1}{\omega \sum_{i=1}^{\omega} (N_i/N) \times \xi_i}, \tag{24}$$

where ω is the number of connected subgraphs of the network, N_i is the number of nodes inside the ith connected subgraph,

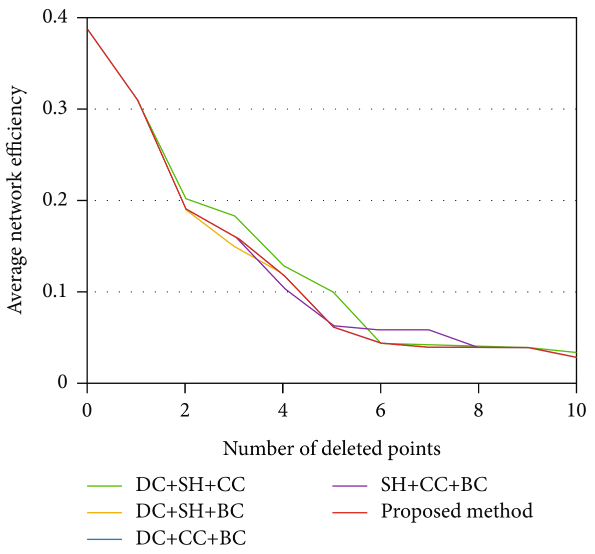


Figure 5: Comparison of different combinations of the three metrics with the method in this study.

and ξ_i and is the average distance inside the ith connected subgraph, expressed as follows:

$$\xi = \frac{1}{N \times (N-1)} \times \sum_{i=1}^{N} \sum_{j=i+1}^{N} d_{ij}. \tag{25}$$

4.1.3. The Maximum-Connected Subgraph Ratio. This is defined as the ratio of the number of nodes in the maximum-connected subgraph in the network to the total number of nodes in the network, expressed as follows:

$$S = \frac{E_m}{E}, \tag{26}$$

where E_m is the number of nodes in the maximum-connected subgraph and E is the total number of nodes in the initial network. The faster the maximum-connected subgraph ratio decreases, the more severely the network is segmented, indicating that the more removed points are important.

4.2. Algorithm Examples

4.2.1. Analysis of the Effectiveness. In order to illustrate the effectiveness of this method, ARPA network is used in this paper. Figure 2 shows the ARPA (Advanced Research Projects Agency) network topology, which consists of 21 nodes and 23 links.

Table 2 gives the results of the ranking of the node importance determined by the algorithm proposed in this study and DC, SH, CC, and BC on the ARPA network.

The method of this study and DC, SH, CC, and BC all have 10 identical nodes with different rankings, showing that the proposed method has certain feasibility. From the overall view of the sorting results, DC, SH, and CC all have different nodes in the same ranking, and it is obviously difficult to distinguish their importance; BC and the proposed method can perform better. From the point of view of a single node,

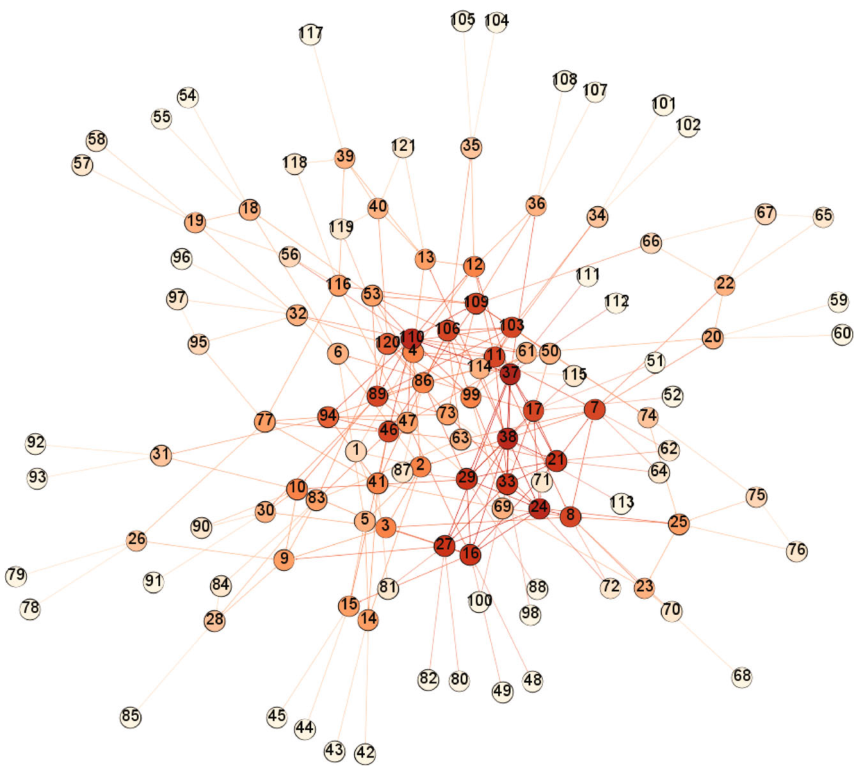


Figure 6: C2 network.

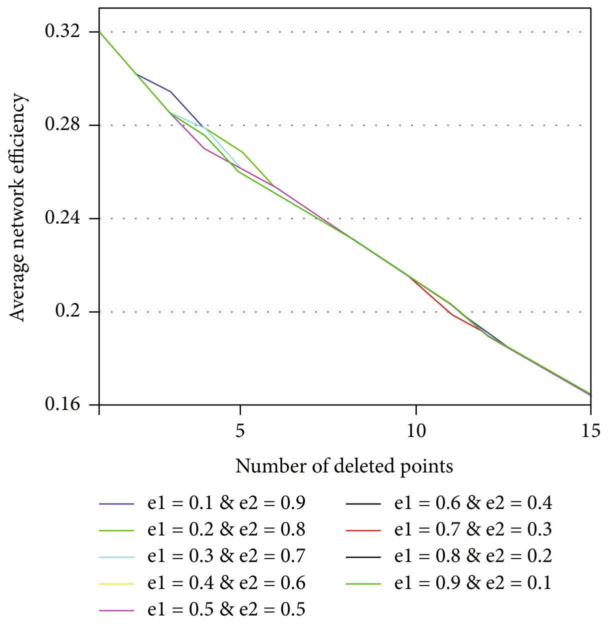


Figure 7: Comparison of different combinations of preference coefficient.

node 2 is more important than node 4 different from CC and BC. As shown in Figure 2, node 2 links with more nodes and plays an important role in information flow, which is obviously more important than node 4, similarly for the comparison between node 12 and node 14.

In order to further illustrate the effectiveness of the algorithm in this paper, the average network efficiency is used for comparative analysis; the importance of nodes is judged by the rate of decline of the average network efficiency after deleting nodes.

It can be seen from the Figure 3 that the average network efficiency of the algorithm in this paper decreases the fastest when the first 2 nodes are deleted, same as BC; explain that node 12 is more important. With the deletion of nodes, the average network efficiency of the algorithm in this paper decreases faster than these four methods, indicating that the above description of a single node is more precise; the algorithm in this study is more reasonable than other algorithms.

4.2.2. Analysis of Different Metric Combinations. To verify the effectiveness of the method itself in this study, the ARPA network was also used for the analysis to verify the effectiveness of the multiattribute fusion method by comparing it with different combinations of individual metric. The results are shown in Figures 4 and 5.

From Figures 4 and 5, the overall decline rate of the average network efficiency of this study's method is higher than the different combination methods of each metric.

When comparing two metric combination methods, this method performs significantly better in removing the first five key nodes and the first ten key nodes than other methods.

Similarly, when comparing three metric combination methods, this method outperforms the others. It is clear that the combination proposed in this study is reasonable and its performance is better than other combination methods.

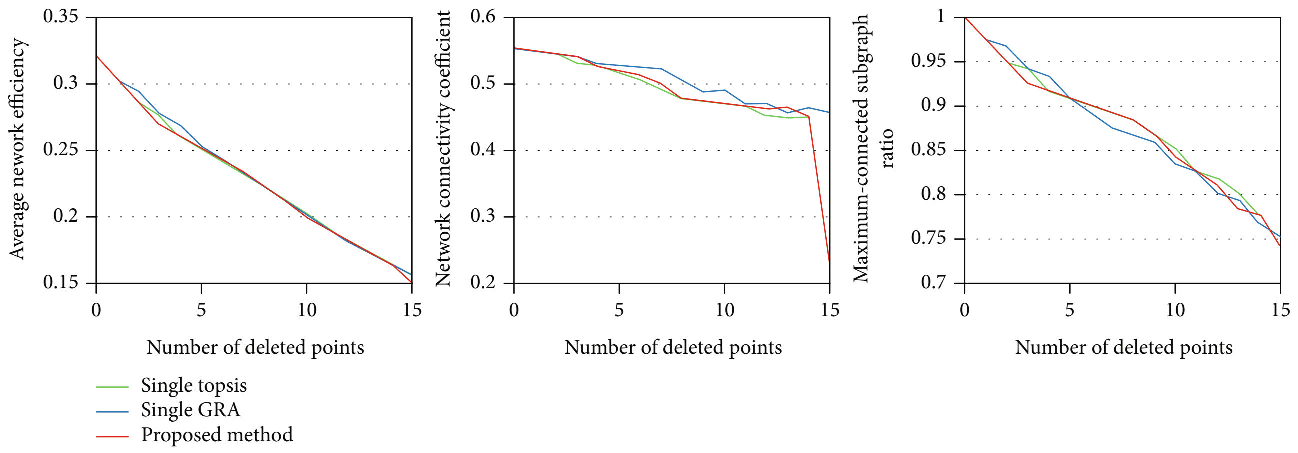


Figure 8: Comparison of different evaluation method.

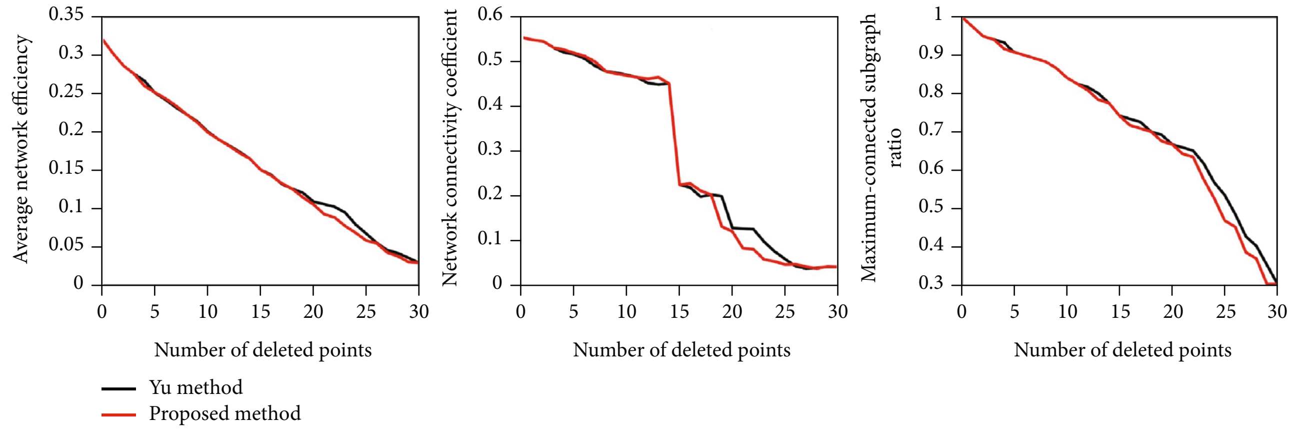


Figure 9: Comparison of different evaluation metrics under the C2 network.

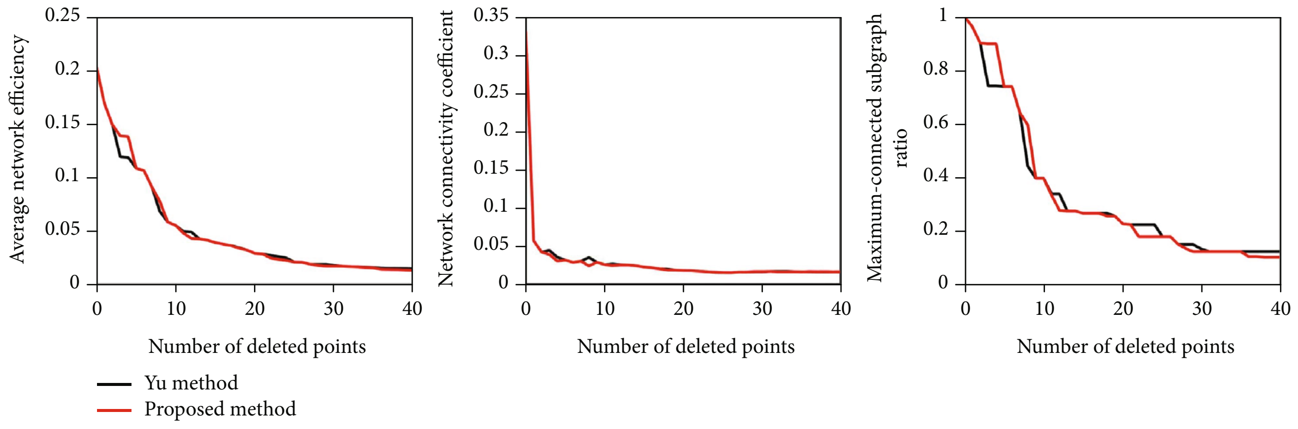


Figure 10: Comparison of different evaluation metrics under the Ca-netscience network.

4.2.3. Analysis of Different Preference Coefficient Combinations. For the setting of the preference coefficient, the influence of the preference coefficient (e_1, e_2) on the experimental results is verified by setting 9 pairs of different combinations. The method of this study is applied to the C2 (command and control) network with 121 nodes, which is a typical air defense command and control system network, and the network is constructed by modeling method. The network structure is shown in Figure 6. The average network efficiency is also used for comparative analysis. The experimental results are shown in Figure 7.

It can be seen from the Figure 7 that the different combinations of preference coefficient have little change in the performance of the network. $e_1 = e_2 = 0.5$ has slightly better

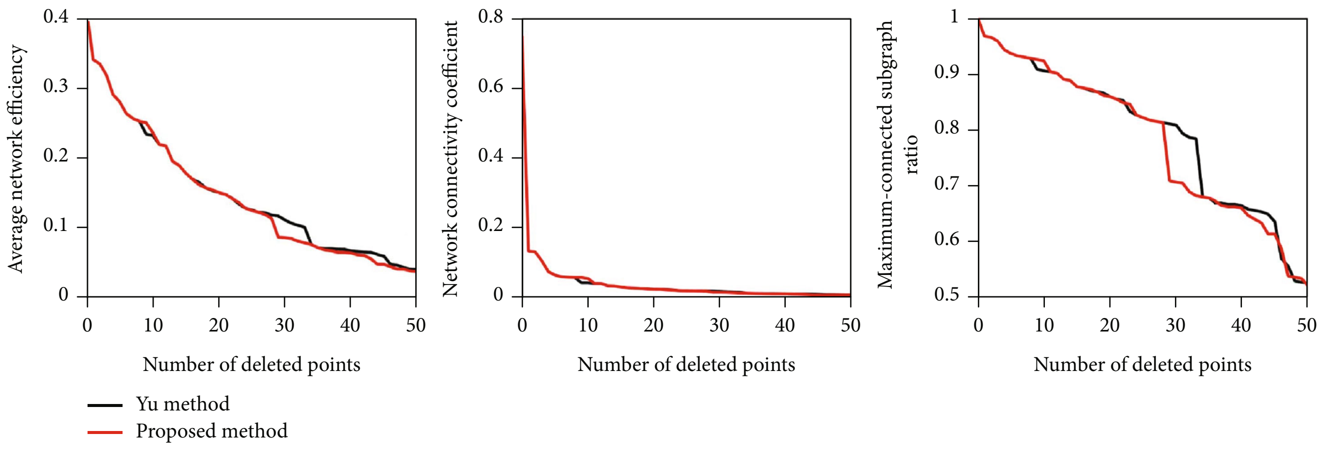


Figure 11: Comparison of different evaluation metrics under the biocelegans network.

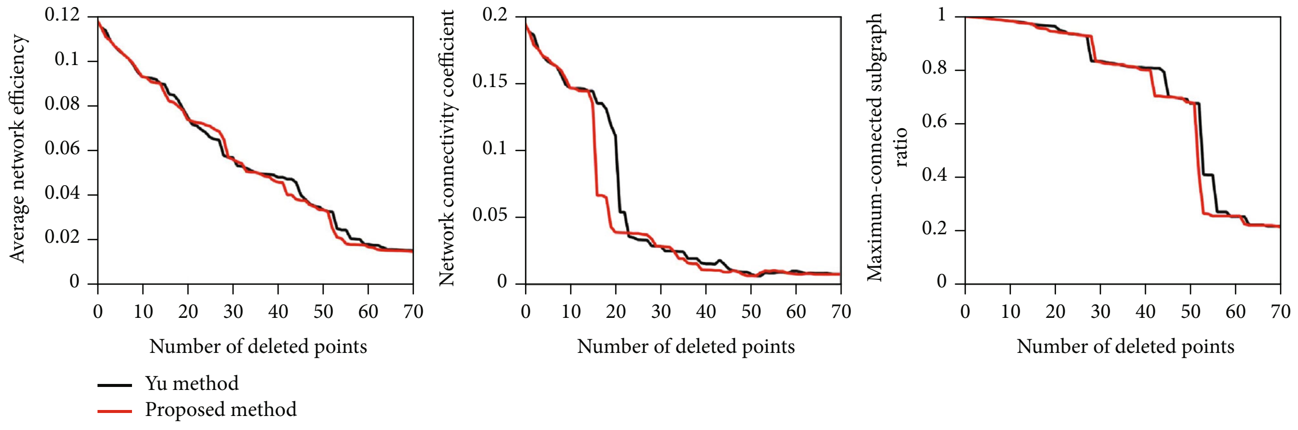


Figure 12: Comparison of different evaluation metrics under the power network.

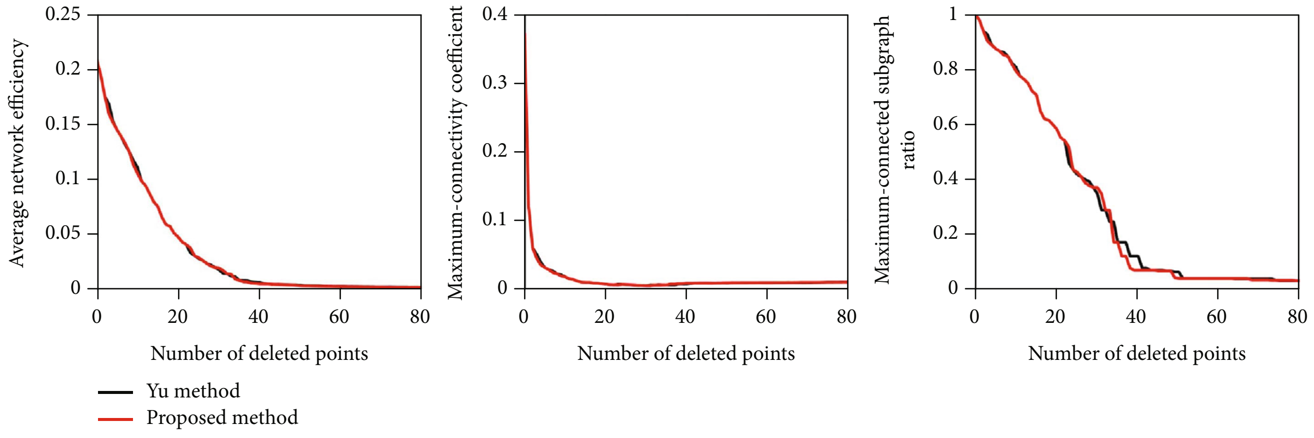


Figure 13: Comparison of different evaluation metrics under the retweet network.

performance and is taken in the subsequent simulations of this paper.

4.2.4. Performance Analysis of Different Evaluation Method. In order to prove the superiority of the evaluation method combining TOPSIS and GRA (grey relational analysis) in this paper, a comparison test between the single evaluation method and the combination method proposed in this paper is designed, and C2 network was used for experiments. The average network efficiency, network connectivity coefficient, and maximum connectivity subgraph ratio are used for comparative analysis. The experimental results are shown in Figure 8.

For the average network efficiency, the performance of the proposed method is slightly better than the single evaluation method. For the network connectivity coefficient, the

Table 3: Statistical characteristics of each network.

Network	Nodes number	Edges number	Attribute
C2	121	256	Air defense command and control system network; nodes are command entities, and edges are abstractions of relationships between entities
Ca-netscience	379	914	Scientific collaboration network in network theory and experiments; nodes are scientists, and edges are cooperative relationship
Biocelegans	453	4600	Metabolic network of celegans; nodes are substrates, edges are metabolic reactions
Power	662	906	Power networks; nodes are power lines, edges are substations
Retweet	761	1000	Retweet and mentions network from the UN conference held in Copenhagen; nodes are twitter users and edges are retweets.

proposed method is slightly inferior to the single TOPSIS, but significantly better than the single GRA, and for the maximum connectivity subgraph ratio, the performance is just the opposite, slightly inferior to a single GRA, but better than a single TOPSIS. In general. The combined method proposed in this paper is feasible, and because it can combine the advantages of the two methods, it performs better than the single evaluation method.

4.2.5. Performance Analysis of Different Networks. To further illustrate the applicability of this method, the method of this study is applied to other different networks including computer generated network C2 network and real-world networks including Ca-netscience network, biocelegans network, power network, and retweet network.

The method in this study is compared with Yu et al.'s method [13], and the evaluation methods used for analysis are network efficiency, network connectivity coefficient, and maximum connectivity subgraph ratio. Because removing 5%-10% of the important nodes in the network is enough to bring down the network, the top ranked nodes are removed in different network. The performance of each method is observed, and the simulation plots are shown in Figures 9–13.

In this paper, five real networks are selected as test networks, and the statistical characteristics of each network are shown in Table 3. Except C2 network constructed by modeling method, other networks are from https://networkrepository.com/

(1) C2 Network. The analysis results of C2 network are shown in Figure 9; for average network efficiency and maximum connectivity subgraph ratio, the performance is slightly better than Yu et al.'s method when the first 15 nodes were deleted. However, this method is significantly better than Yu et al.'s method after the 15th node is deleted. For network connectivity coefficient, the performance of this method is slightly inferior to the method in some periods, and the method in this paper is still improved in general.

(2) Ca-netscience Network. In Figure 10, for average network efficiency and maximum connectivity subgraph ratio, the performance is slightly inferior to Yu et al.'s method when the first 10 nodes were deleted, but the effect of this paper is obviously better than after the 10th node is deleted. For network connectivity coefficient, our method outperforms Yu et al.'s method in the whole process. On the whole, our method performs slightly better than Yu et al.'s method.

(3) Biocelegans Network. As shown in Figure 11, the performance of our method in this study is slightly inferior to Yu et al.'s method when the first 10 nodes were deleted, but after that, the effect of this method is obviously better in different evaluation methodology.

(4) Power Network. In Figure 12, for average network efficiency, the performance of our method in this paper is slightly worse than Yu et al.'s method when deleting 20th nodes to 30th nodes, and others period perform better. For network connectivity coefficient and maximum connectivity subgraph ratio, our method works significantly better.

(5) Retweet Network. As shown in Figure 13, for retweet network, the performance of the approach in this paper is almost identical to Yu et al.'s method. But for maximum connectivity subgraph ratio, the improvement effect of our method in this paper is more obvious.

We notice that the method in this paper is slightly worse in some periods; the reason for this phenomenon is related to the network structure of the network itself. The basic evaluation metrics selected in this study are not quite suitable for these networks, which leads to the difference in the evaluation performance. It is necessary to select more suitable metrics for the structure of the different network, which is one of the future research directions. But in general, our method is clearly more suitable for various networks and performs better in different evaluation methods.

In summary, this method shows better performance in different combinations of node importance evaluation and has good algorithmic applicability to be applied in different networks, which has some practical value.

5. Conclusions

In this study, a subjective and objective comprehensive weighting method is proposed to weight the decision matrix, and combined with the advantages of TOPSIS and grey relational analysis algorithm, the relative closeness is proposed and applied to the node importance identification of complex networks. Finally, different comparative experiments and evaluation methodology are designed to analyze the performance of the algorithm. For the effectiveness of the algorithm itself in this study, its performance outperforms the combined approach with different metrics. Furthermore, this algorithm outperforms Yu et al.'s method in terms of average network efficiency, network connectivity coefficient, and maximum connectivity subgraph ratio for different types of networks, which indicates that this scheme is more reasonable and the evaluation results of nodes are closer to the actual situation and achieve good results.

The paper also has potential limitations, such as the selection of indicators in different networks, AHP will cause rank inversion problems, and whether the algorithm is still applicable in dynamic network link prediction. As future research, look for new weight calculation methods, such as alpha-discounting method to solve the problem [21], and try to use LSTM [22, 23] to solve the link prediction problem.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by Natural Science Foundation of Guangdong Province (2022A1515010737) and China University Industry-University-Research Collaborative Innovation Fund (2021FNA04017).

References

[1] Q. C. Hu, Y. S. Yin, P. F. Ma, Y. Gao, Y. Zhang, and C. X. Xing, "A new approach to identify influential spreaders in complex networks," *Acta Physica Sinica*, vol. 62, no. 14, article 140101, 2013.

[2] B. L. Zhang, *Research on Information Diffusion and Influence Maximization in Social Networks*, Nanjing University, 2016.

[3] Y. S. Jiang, "Key node identification of navigation network based on structural centrality," *Computer and Modernization*, no. 7, pp. 108–113, 2018.

[4] K. Yan, L. Li, and Y. B. Qin, "Key node identification methods based on road traffic networks," *Computer Engineering & Science*, vol. 40, no. 11, pp. 1983–1990, 2018.

[5] P. Ren, C. Li, P. Tao, C. F. Wu, and H. Li, "Node vulnerability evaluation for power network based on weighted entropy TOPSIS method," *Journal of Electric Power Science and Technology*, vol. 34, no. 3, pp. 143–149, 2019.

[6] G. H. Zhao, P. Jia, and A. M. Zhou, "Improved degree centrality for directed-weighted network," *Journal of Computer Applications*, vol. 40, Supplement 1, pp. 141–145, 2020.

[7] M. M. Tulu, R. Hou, and T. Younas, "Identifying influential nodes based on community structure to speed up the dissemination of information in complex network," *IEEE Access*, vol. 6, pp. 7390–7401, 2018.

[8] K. L. Wang, C. X. Wu, J. Ai, Z. Su, and School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China, "Complex network centrality method based on multi-order K-shell vector," *Acta Physica Sinica*, vol. 68, no. 19, article 196402, 2019.

[9] Y. M. Wang, Q. Y. Wang, C. S. Pan, and B. Chen, "Method for key nodes identification in command and control network by considering structural holes," *Fire Control & Command Control*, vol. 42, no. 3, pp. 59–63, 2017.

[10] Y. Wang, S. Wang, and Y. Deng, "A modified efficiency centrality to identify influential nodes in weighted networks," *Pramana*, vol. 92, no. 4, 2019.

[11] G. Hu, X. Xu, H. Gao, and X. C. Guo, "Node importance recognition algorithm based on adjacency information entropy in networks," *Systems Engineering-Theory & Practice*, vol. 40, no. 3, pp. 714–725, 2020.

[12] S. Xu and P. Wang, "Identifying important nodes by adaptive LeaderRank," *Physica A Statistical Mechanics & Its Applications*, vol. 469, pp. 654–664, 2017.

[13] H. Yu, Z. Liu, and Y. J. Li, "Key nodes in complex networks identified by multi-attribute decision-making method," *Acta Physica Sinica*, vol. 62, no. 2, article 020204, 2013.

[14] Z. Liu, C. Jiang, J. Wang, and H. Yu, "The node importance in actual complex networks based on a multi-attribute ranking method," *Knowledge-Based Systems*, vol. 84, pp. 56–66, 2015.

[15] Y. Z. Yang, L. Yu, X. Wang, S. Y. Chen, Y. Chen, and Y. P. Zhou, "A novel method to identify influential nodes in complex networks," *International Journal of Modern Physics C*, vol. 31, no. 2, p. 2050022, 2020.

[16] J. T. Hu, Y. X. Du, H. M. Mo, D. J. Wei, and Y. Deng, "A modified weighted TOPSIS to identify influential nodes in complex networks," *Physica A Statistical Mechanics & Its Applications*, vol. 444, pp. 73–85, 2016.

[17] P. L. Yang, X. Liu, and G. Q. Xu, "A dynamic weighted TOPSIS method for identifying influential nodes in complex networks," *Modern Physics Letters B*, vol. 32, no. 19, p. 1850216, 2018.

[18] X. L. Yan, Y. P. Cui, and S. J. Ni, "Identifying influential spreaders in complex networks based on entropy weight method and gravity law," *Chinese Physics B*, vol. 29, no. 4, article 048902, 2020.

[19] X. Liu, G. Q. Xu, and P. L. Yang, "Node importance evaluating of network based on combination weighting VIKOR method," *Application Research of Computers*, vol. 36, no. 8, pp. 2368–2371, 2019.

[20] T. Bian and Y. Deng, "A new evidential methodology of identifying influential nodes in complex networks," *Chaos Solitons & Fractals*, vol. 103, pp. 101–110, 2017.

[21] F. Smarandache, "Three non-linear a-discounting MCDM-method examples," in *Proceedings of The 2013 International Conference on Advanced Mechatronic Systems (ICAMechS 2013)*, pp. 174–176, Luoyang, China, 2013.

[22] L. Zhang, Z. Huang, W. Liu, Z. Guo, and Z. Zhang, "Weather radar echo prediction method based on convolution neural network and long short-term memory networks for sustainable e-agriculture," *Journal of Cleaner Production*, vol. 298, article 126776, 2021.

[23] L. Zhang, C. Xu, Y. Gao, Y. Han, X. du, and Z. Tian, "Improved Dota2 lineup recommendation model based on a bidirectional LSTM," *Tsinghua Science and Technology*, vol. 25, no. 6, pp. 712–720, 2020.

Hindawi
Wireless Communications and Mobile Computing
Volume 2023, Article ID 2854874, 15 pages
https://doi.org/10.1155/2023/2854874

Research Article

KC-GCN: A Semi-Supervised Detection Model against Various Group Shilling Attacks in Recommender Systems

Hongyun Cai,[1,2] **Jichao Ren**,[1,2] **Jing Zhao**,[3] **Shilin Yuan**,[1,2] **and Jie Meng**[1,2]

[1]*School of Cyber Security and Computer, Hebei University, Baoding 071000, China*
[2]*Key Laboratory on High Trusted Information System in Hebei Province, Hebei University, Baoding 071000, China*
[3]*Security Department, Hebei University, Baoding 071000, China*

Correspondence should be addressed to Jichao Ren; renjich@163.com

Received 7 July 2022; Revised 7 November 2022; Accepted 1 February 2023; Published 16 February 2023

Academic Editor: Yawen Chen

Various detection methods have been proposed for defense against group shilling attacks in recommender systems; however, these methods cannot effectively detect attack groups generated based on adversarial attacks (e.g., GOAT) or mixed attack groups. In this study, we propose a two-stage method, called KC-GCN, which is based on k-cliques and graph convolutional networks. First, we construct a user relationship graph, generate suspicious candidate groups, and extract influential users by calculating the user nearest-neighbor similarity. We construct the user relationship graph by calculating the edge weight between any two users through analyzing their similarity over suspicious time intervals on each item. Second, we combine the extracted user initial embeddings and the structural features hidden in the user relationship graph to detect attackers. On the Netflix and sampled Amazon datasets, the detection results of KC-GCN surpass those of the state-of-the-art methods under different types of group shilling attacks. The F1-measure of KC-GCN can reach above 93% and 87% on these two datasets, respectively.

1. Introduction

The amount of Internet data is exploding with the rapid development of information technology, consequently leading to the increasingly prominent problem of information overload. By analyzing a user's historical behavior information, recommender systems can extract user preferences and automatically recommend favorite items or services to users [1–3], which have become an essential component of many online information services, including e-commerce [4, 5], live broadcast platforms [6], personalized travel recommendation systems [7], and Internet of Vehicles wireless systems [8], among many others. However, due to their openness, fraudulent users can create and inject a large number of fake user profiles into recommender systems, which can change recommendation results and reduce user experience. For example, The New York Times and Buzzfeed News have reported that many sellers turned to black hat tactics to drive Amazon sales on their products (https://pattern.com/news/pattern-analysis-on-amazon-star-rating-featured-in-new-york-times-buzzfeed/). In recent years, various types of shilling attack models have been presented, including random attacks [9], average attacks [10], and the latest adversarial attacks [11]. Group shilling attacks have also been proposed to generate a group of attack profiles on the basis of the abovementioned individual shilling attacks [12]. Research on group shilling attacks showed that group attacks greatly affect recommender systems when compared to traditional individual attacks [13, 14] because attack users in the same shilling group collude with each other to attack targets, while each attack profile looks more like a genuine profile [15]. Nowadays, people have become increasingly conscious of the importance of shilling attack governance in recommender systems. Many service platforms, such as Amazon, Tripadvisor, and Taobao, are constantly seeking efficient mechanisms to enhance user experience and satisfaction (https://www.bbc.com/news/business-61154748). Therefore, accurate detection under group shilling attacks has emerged as a crucial problem for the existing recommender system security.

In recent years, various detection approaches have been put forward to defend recommender systems from group shilling attacks [16–21]. Most of these methods detect shilling groups based on frequent synchronization behaviors on more than one item or through the analysis of the differences in the rating pattern of genuine and attack users. However, these approaches do not perform well if the group attack profiles are generated and injected based on AOP [9], adversarial attacks, or mixed attacks because all attackers in the same shilling group may not attack identical target items. The injected attack profiles are also diverse and look more like genuine ones under these attacks.

To solve the abovementioned constraints, we present herein a two-stage method, called KC-GCN. It is a semi-supervised group shilling attack detection model based on k-cliques and the graph convolutional network (GCN) [22, 23]. First, a user relationship graph is generated, and influential users are extracted using the k-clique algorithm and the user nearest-neighbor similarity on the graph. We construct the user relationship graph by calculating the edge weight between any two users through the analysis of their similarity over suspicious time intervals on each item. Second, we obtain the user initial embeddings and train a GCN-based classifier.

The significant contributions of this work are as follows:

(1) We construct a weighted user relationship graph, in which the weight is calculated from the perspectives of user preference, attack intention, and time synchronization, to highlight the user relationship between attack users

(2) We use the multilayer graph convolution network to fuse the initial embedded features extracted from the user rating behavior with the structural features hidden in the user relationship graph to extract more effective detection features

(3) The experiments on the Netflix and Amazon datasets demonstrate that KC-GCN outperforms baseline methods in terms of detecting various types of group shilling attacks

The rest of this paper is structured as follows: Section 2 presents the background information and related work, Section 3 provides a detailed description of the proposed detection methodology primarily divided into two sections (i.e., extracting influential users and identifying attack users using the trained semi-supervised classifier), Section 4 provides a comparative analysis of the experimental findings, and Section 5 presents the conclusions.

2. Background and Related Work

2.1. Group Shilling Attacks. To escape from the existing methods of detecting individual shilling attacks (e.g., random attack, average attack, and AOP attack), Wang et al. [24] proposed two generative models of group shilling attack, called $\mathrm{GSAGen_s}$ and $\mathrm{GSAGen_l}$. In these attack models, fake profiles are first generated based on one type of individual shilling attacks. Based on which, group shilling attack profiles are then constructed and injected into a set of genuine profiles. The $\mathrm{GSAGen_s}$ model has more stringent conditions when generating group shilling profiles; hence, the group size under $\mathrm{GSAGen_s}$ is smaller than that under $\mathrm{GSAGen_l}$. Considering the attack effect on the target items, we only use $\mathrm{GSAGen_l}$ to generate the group shilling attack profiles, in which the fake profile includes the selected item set, the filler item set, the target item set, and the unrated item set. More details for the group shilling attacks used in this paper are described as follows:

(1) *$GSAGen_l$ Ran*: generate loose group attack profiles based on a random attack, where the selected items are null, the filler items are randomly chosen, and only one attacker from the whole group rates the items. The filler item rating is the system mean. The target item rating is set to $r_{\max}$ or $r_{\min}$

(2) *$GSAGen_l$ Ave*: generate loose group attack profiles on the basis of an average attack, where the selected items are null, the filler items are randomly chosen, and only one attacker from the whole group rates the items. The filler item rating is the item mean. The target item rating is set to $r_{\max}$ or $r_{\min}$

(3) *$GSAGen_l$ AOP*: generate loose group attack profiles based on 50% AOP attack, where the selected items are null, the filler items are randomly chosen, and only one attacker from the whole group rates those items with top 50% popularity. The filler item rating is the item mean. The target item rating is set to $r_{\max}$ or $r_{\min}$

(4) *$GSAGen_l$ GOAT*: generate loose group attack profiles based on the adversarial attack, called GOAT [11], where each fake user's selected and filler items are randomly chosen from an item-item graph based on genuine user profiles. A generative adversarial network is used to generate the ratings of the selected and filler items based on the genuine rating distribution. The target items have ratings of $r_{\max} - 1$ or $r_{\min} + 1$

(5) *$GSAGen_l$ Mixed*: generate mixed multiple shilling groups generated according to the four abovementioned group shilling attacks

2.2. Related Work

2.2.1. Individual Shilling Attack Detection Methods. Chirita et al. [25] and Burke et al. [26] proposed attack user detection indicators to identify shilling attackers based on fraudulent user rating behavior patterns. These indicators were suitable for detecting specific attacks (i.e., random attack), but failed under obfuscated attacks (e.g., AOP attack). To detect various attack types, Zhang et al. [27] proposed an attack detection framework based on label propagation. The framework used the label propagation algorithm to obtain the suspicious probability of each user. Although this method did not require the consideration of particular attack strategies, it needs a certain number of seed users and must know the number of attackers in advance. Zhang et al. [28]

put forward an unsupervised attack detection method, called UD-HMM, which first determined each user's degree of suspicion based on their hidden Markov model behavior before utilizing hierarchical clustering to identify attackers. However, this method did not work for detecting the AOP attack profiles. Yang et al. [29] proposed a unified detection framework that can detect various malicious attacks, including common access injection and shilling attacks. Their framework transformed the user rating behavior into a coupled association network. The network connections and nodes were assessed for trustworthiness by utilizing coupling factor graphs and label propagation algorithms. Meanwhile, Hao and Zhang [30] proposed a deep learning-based and community detection unsupervised approach, called DECDM. They constructed a graph of weighted user relationships based on user behavior similarity and then reconstructed the user relationship graph using stacked denoising autoencoders (SDAEs) and the k-means algorithm. The experiments showed that the method has excellent detection performance on multiple individual shilling attacks. However, it uses the SDAEs multiple times to extract graph features with different damage rates, resulting in a high algorithm time complexity. Ebrahimian and Kashef [31] proposed a hybrid shilling attack detection model based on the convolutional and recurrent neural networks, which first converted the rating matrix into a three-dimensional array of users, products, and days; extracted the user feature vector by using the convolutional neural network (CNN) model; and finally used the RNN model to divide users into two categories: genuine and attacker users. This model did not rely on specific types of attacks and considered the user characteristics in the time dimension. However, the experimental results on the two datasets of Netflix and MovieLens showed that the detection performance was extremely unstable as the filler size changed. Zhou and Duan [32] proposed a coforest algorithm-based semi-supervised recommendation attack detection method that requires setting a reasonable value for each hyperparameter. Zhang et al. [33] proposed GraphRfi, which trains the GCN to obtain the prediction error and introduces neural random forests to detect fraudulent users. Similar to that in [32], the method also requires multiple hyperparameters, and the detection result is easily affected by the hyperparameters.

2.2.2. Group Attack Detection Methods. Zhou et al. [16] proposed the DeR-TIA to identify group attack profiles. In the first stage, they calculated the user profile attributes using improved RDMA and DegSim. In the second stage, they filtered out attack profiles by using the target item analysis. This method works well for identifying high-correlation attack profiles but fails to detect attack groups with a strong diversity. Zhou et al. [17] proposed a detection method, called SVM-TIA, based on the support vector machine and target item analysis. This method can improve the detection precision by using target items but does not have a high recall. Zhang and Wang [18] proposed the GD-BKM method to detect group shilling attacks. They generated candidate groups based on the rating tracks for each item and calculated the candidate group suspiciousness using the user activity and group item attention degree. They then finally spotted attack groups by using the bisecting k-means algorithm. This method can exhibit an excellent detection performance, regardless of the number of target items. However, it becomes less effective when the size of the shilling group is small. Zhang et al. [19] proposed the GAGE method based on graph embedding. First, they extracted user embeddings using the Node2vec method. Next, they obtained candidate groups by using the k-means++ algorithm and calculated the group suspicious degrees. Ultimately, they identified attack groups using Ward's hierarchical-clustering algorithm. Their method uses Node2vec sampling with a certain randomness, thereby leading to deviations in the candidate group division and unstable detection results. Meanwhile, Yu et al. [20] proposed the GAD-MDST method based on maximum dense subtensor mining. This method can automatically generate multiscale user features by fusing a CNN and a feature pyramid network but is not suitable for detecting smaller-sized shilling groups. In our previous work [21], we proposed the TP-GBF method by using strongly correlated behaviors among group members and group behavior characteristics, which combined indirect behaviors with the direct collusion behaviors to highlight the collusive relationship between attackers in the same shilling group. TP-GBF performed well on the Netflix dataset but was less effective on the real dataset because it failed to detect smaller-sized attack groups.

For easy comparison of the above works, we summarize them in Table 1.

3. GCN-Based Group Shilling Attack Detection Model

Figure 1 depicts the two stages of the KC-GCN detection framework: influential user extraction and attack user identification. In the first stage, we build the user relationship graph by determining the user similarity based on the item suspicious time window. Next, we use the k-clique community discovery algorithm to generate suspicious candidate groups. Finally, we obtain the influential users by calculating the user nearest-neighbor similarity. In the second stage, we extract the users' initial embeddings from four dimensions. We then combined the extracted user initial embeddings with the structural features hidden in the user relationship graph to train a semi-supervised classifier based on a two-layer GCN, which only utilizes the labels of the identified influential users.

The notations used in this paper are described in Table 2.

3.1. Extracting Influential Users

3.1.1. Constructing a Weighted User Relationship Graph. Attackers in a shilling group typically cooperate to quickly enhance or demote the recommendation of one or more target items. Based on this characteristic of group attacks, the rating distribution of a target item may fluctuate during the attacked time period. Therefore, we construct a weighted user relationship graph by extracting the suspicious time windows of the suspicious items and calculating the correlation between users within the suspicious time windows.

Table 1: Comparison of different shilling attack detection methods.

Category	Approaches	Advantage	Disadvantage
Individual shilling attack detection methods	Chirita et al. [25] and Burke et al. [26]	Effective for specific attacks	Less effective under obfuscated attacks, e.g., AOP attack
	Zhang et al. [27]	A unified framework for detecting various shilling attacks	Require prior knowledge of attacks
	Zhang et al. [28]	Effective for a wide variety of shilling attacks	Less effective under the AOP attack
	Yang et al. [29]	A unified framework for detecting common access injection and shilling attacks	Require setting more parameters
	Hao and Zhang [30]	Automatic feature learning	High computational cost
	Ebrahimian and Kashef [31]	Regardless of the specific attacks	Unstable detection performance
	Zhou and Duan [32]	High detection precision	Require setting hyperparameters
	Zhang et al. [33]	Consider both user preference and reliability	Require multiple hyperparameters
Group attack detection methods	Zhou et al. [16]	Effective for detecting those shilling group profiles with high-correlation	Less effective for attack group profiles with a strong diversity
	Zhou et al. [17]	High detection precision	Low recall under attacks with a small attack size
	Zhang and Wang [18]	Excellent detection performance regardless of the number of target items	Less effective under smaller-sized groups
	Zhang et al. [19]	Automatic feature extraction	Unstable detection results
	Yu et al. [20]	Automatic feature extraction	Less effective under a smaller group size
	Cai and Zhang [21]	Effective for detecting tightly coupled shilling groups	Less effective under smaller-sized shilling groups on the Amazon dataset

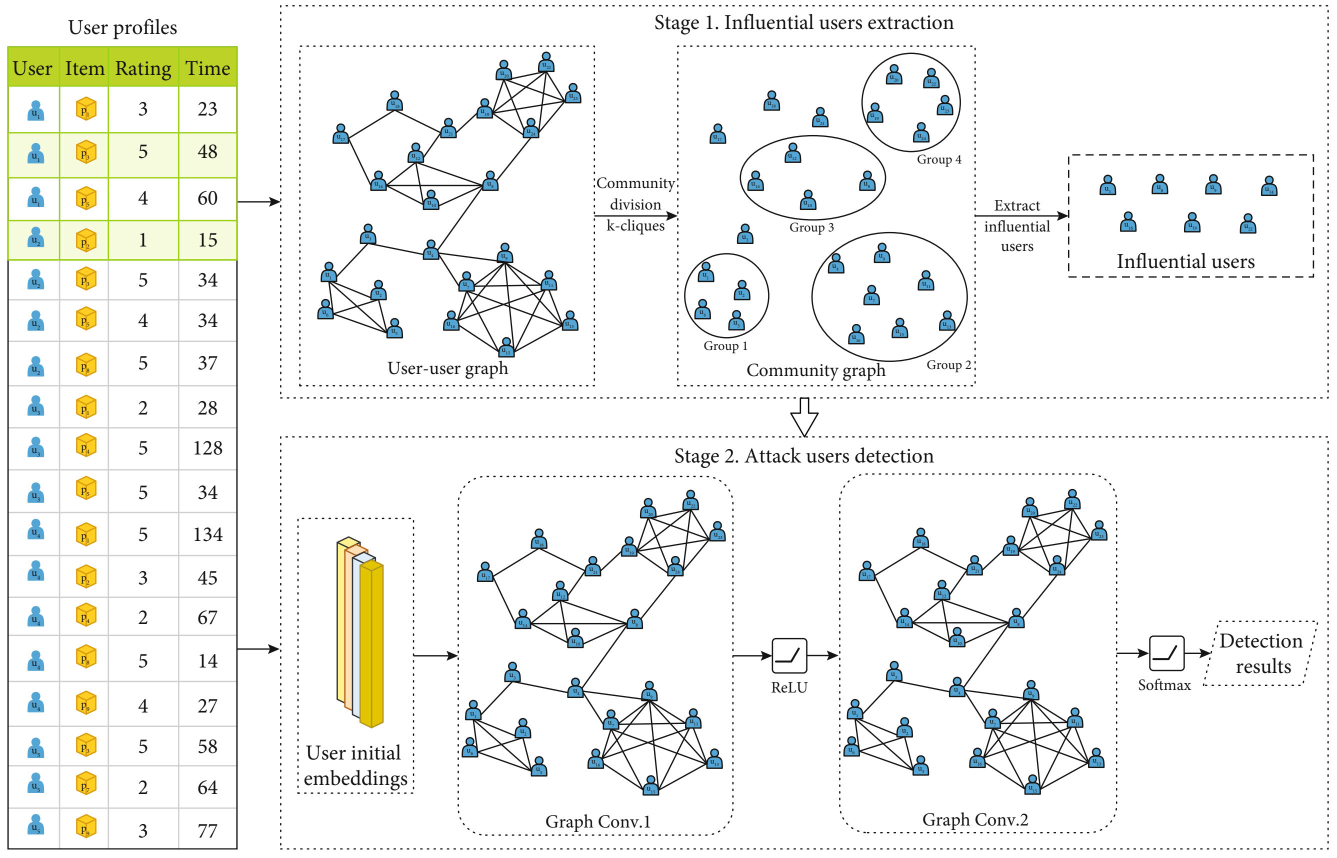


Figure 1: Detection framework of KC-GCN.

Table 2: Notations and their descriptions.

Notation	Description
$U = \{u_1, u_2, \cdots, u_m\}$	Set of users in the rating dataset
$P = \{p_1, p_2, \cdots, p_n\}$	Set of items in the rating dataset
$R = [r_{ij}]_{m\times n}$	User-item rating matrix
$T = [t_{ij}]_{m\times n}$	User-item rating time matrix
$W = \{w_1, w_2, \cdots, w_v\}$	Set of time windows in the rating dataset
$G = \langle U, E, G\rangle$	A weight user relationship graph
$\lvert\cdot\rvert$	The number of elements in a set
X	Users' initial embedding matrix

Definition 1 (item window abnormal degree, $IWAD$). For $\forall p \in P$ and $\forall w \in W$, the abnormal degree of item p on window m refers to the ratio of the number of users who rated item p with high ratings to the total number of users who rated it on time window w, which is referred to $\mathrm{IWAD}_{p,w}$ and calculated by

$$\mathrm{IWAD}_{p,w} = \frac{\sum_{u\in U}\Gamma(u, p, w)}{\mathrm{NR}_{p,w}}, \tag{1}$$

where $\mathrm{NR}_{p,w}$ represents the number of ratings of item p on the time window w. The time window is regarded as suspicious if $\mathrm{IWAD}_{p,w} > 0.5$. $\Gamma_{(u,p,w)}$ is an indicator function, which is formulated as

$$\Gamma_{u,p,w} = \begin{cases} 1, & \text{if } r_{u,p} \geq 4, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Definition 2 (user rating synchronization, URS). For $\forall u_i, u_j \in U$, their rating synchronization refers to how close their rating behavior is within the suspicious time window, which is denoted as $\mathrm{URS}(u_i, u_j)$ and calculated by

$$\mathrm{URS}(u_i, u_j) = \sum_{p\in N(u_i,u_j),r(u_i,p)\geq 4,r(u_j,p)\geq 4}\left(1 - \frac{t(u_i,p) - t(u_j,p)}{\tau}\right), \tag{3}$$

where $N(u_i, u_j)$ represents the set of items corated by users u_i and u_j, and the rating time is within the suspicious time window of item p;that is, $\mathrm{IWAD}_{p,w} > 0.5$.

Definition 3 (user short preference similarity, USPS). For $\forall u_i, u_j \in U$, their short preference similarity is defined as the ratio of $|N(u_i, u_j)|$ to $|N(u_i)| \cup |N(u_j)| - |N(u_i, u_j)|$, which is denoted as $\mathrm{USPS}(u_i, u_j)$ and calculated by

$$\mathrm{USPS}(u_i, u_j) = \frac{|N(u_i, u_j)|}{|N(u_i)| \cup |N(u_j)| - |N(u_i, u_j)|}, \tag{4}$$

where $N(u_i)$ and $N(u_j)$ represent the rating item set of users u_i and u_j, respectively. $|N(u_i, u_j)|$ represents the number of items for which user u_i and user u_j have the same preference within the suspicious time window, and $|N(u_i)| \cup |N(u_j)|$ represents the total number of items rated by user u_i and user u_j.

Definition 4 (user similarity, US). For $\forall u_i, u_j \in U$, their user similarity refers to the closeness of their rating times and similarity of their preferences on suspicious items, which is denoted as $\mathrm{US}(u_i, u_j)$ and calculated by

$$\mathrm{US}(u_i, u_j) = \mathrm{URS}(u_i, u_j) \times \mathrm{USPS}(u_i, u_j). \tag{5}$$

Based on the above definition, a weighted user relationship graph can be constructed. The weighted user relationship graph construction algorithm is described as follows.

Algorithm 1 is divided into two parts. The first part (lines 1–6) calculates the suspicious time window for each item, with a time complexity of $O(n * v)$. The second part (lines 7–21) calculates the relevance degree US of each user and constructs a user relationship graph, with a time complexity of $O(m^2)$. In conclusion, Algorithm 1 has a time complexity of about $O(m^2)$.

3.1.2. Extracting Influential Users. Li et al. [34] proposed the maximization problem that is aimed at selecting seed nodes from numerous nodes, thereby maximizing the influence of information on large-scale network transmission [35]. Inspired by the seed node idea, we only used the influential node labels to reduce the cost of labeling a large number of samples.

We present herein a two-stage method for extracting influential users. First, we generate suspicious candidate groups on the weighted user relationship graph using the k -clique algorithm [21]. Next, we extract influential users by calculating the user nearest-neighbor similarity.

The main steps of generating candidate groups based on the k-clique algorithm are as follows:

(1) Traverse each node in the user relationship graph to find a complete subgraph $G_i = \{u_1^i, u_2^i, \cdots, u_k^i\}$ containing k users, and add the users in G_i into the tightness community set TCS

(2) Convert the TCS into an overlapping community matrix O, where the diagonal elements in the matrix O represent the number of users in the community, and the off-diagonal elements represent the number of shared users in adjacent communities

(3) Merge the small communities in the overlapping community matrix O to obtain the community adjacency matrix A. In the matrix O, these diagonal elements with a value less than k and off-diagonal elements with a value less than $k - 1$ are set to 0, while the left elements are set to 1

(4) Generate the suspicious candidate group based on the community adjacency matrix A

```
Input: the rating matrix $R$, the rating time matrix $T$, the size of sliding time window $W_s$, the time window anomaly threshold $\delta$, and
the relationship strength threshold $\sigma$
Output: a weighted user relationship graph $G$
1.  $E \longleftarrow \varnothing$ ; $C \longleftarrow 0_{m\times n}$
2.  for each item $p \in P$ do
3.    for each time window $\forall w \in W$ do
4.      compute $\mathrm{IWAD}_{p,w}$ according to Eq. (1)
5.    end for
6.  end for
7.  for each user $u_i \in U$ do
8.    for each user $u_j \in U$ do
9.      compute $\mathrm{URS}(u_i, u_j)$ according to Eq. (3)
10.       compute $\mathrm{USPS}(u_i, u_j)$ according to Eq. (4)
11.       if $\mathrm{URS}(u_i, u_j) > \sigma$ then
12.         compute $\mathrm{US}(u_i, u_j)$ according to Eq. (5)
13.         $C[u_i][u_j] \longleftarrow \mathrm{US}(u_i, u_j)$
14.         $E \longleftarrow E \cup \{(u_i, u_j)\}$
15.       else
16.         $\mathrm{US}(u_i, u_j) \longleftarrow 0$
17.       end if
18.     end for
19.   end for
20.   construct a weight user relationship graph $G = \langle U, E, C\rangle$
21.   return $G$
```

Algorithm 1: Constructing a weighted user relationship graph.

Definition 5 (user nearest neighbor similarity, *UNNS*). For $\forall u_i \in U$, the user's nearest neighbor similarity refers to the average similarity between the user and its neighbors, which is denoted as $\mathrm{UNNS}(u_i)$ and calculated by

$$\mathrm{UNNS}(u_i) = \frac{\sum_{u_j \in \mathrm{Neighbor}(u_i)} \mathrm{UR}(u_i, u_j)}{|\mathrm{Neighbor}(u_i)|}, \tag{6}$$

where $UR(u_i, u_j)$ represents the similarity of users u_i and u_j.

Definition 6 (influential user, IU). Influential users refer to those users whose nearest neighbor similarity is larger than that of all its first-order neighbors.

The algorithm for extracting influential users based on the k-clique algorithm and user nearest neighbor similarity is described as follows.

Algorithm 2 is divided into four parts. The first part (lines 1–12) identifies tight communities in the graph and generates a community relationship matrix with a time complexity of $O(m * l) + O(l^2)$. The second part (lines 13–19) merges communities to generate a community adjacency matrix with a time complexity of $O(l^2) + O(l^2)$. The third part (lines 20–24) generates candidate suspicious groups according to the community adjacency matrix, with a time complexity of $O(1)$. The last part (lines 25–29) extracts an influential user set based on the user nearest neighbor similarity, with a time complexity of $O(l * m)$. In conclusion, Algorithm 2 has a time complexity of about $O(m * l)$.

3.2. Detecting Attack Users

3.2.1. Generating User Initial Embeddings.

Some node-embedding methods (e.g., matrix factorization and autoencoders) are automatic but usually generated using a randomization strategy and cannot represent well the initial node embeddings. Therefore, we generate the user embeddings herein from four perspectives.

Definition 7 (user lifetime proportion, ULP). For $\forall u_i \in U$, the user lifetime proportion refers to the ratio of the lifetime of user u_i in the system to the lifetime of the entire system, which is denoted as $\mathrm{ULP}(u_i)$ and calculated by

$$\mathrm{ULP}(u_i) = \frac{\mathrm{URT}(u_i, \max) - \mathrm{URT}(u_i, \min)}{\mathrm{SL}}, \tag{7}$$

where $\mathrm{URT}(u_i, \max)$ and $\mathrm{URT}(u_i, \min)$ represent the latest and the earliest rating time of user u_i, respectively. SL represents the lifetime of the entire system.

Definition 8 (user nearest neighbor rating synchronization, UNNRS). For $\forall u_i \in U$, the user's neighbor rating synchronization refers to the average rating synchronization between the user and its neighbors, which is denoted as $\mathrm{UNNRS}(u_i)$ and calculated by

$$\mathrm{UNNRS}(u_i) = \frac{\sum_{u_j \in \mathrm{Neighbor}(u_i)} \mathrm{URS}(u_i, u_j)}{|\mathrm{Neighbor}(u_i)|}, \tag{8}$$

where $\mathrm{URS}(u_i, u_j)$ means the synchronization degree of user

```
Input: the user's relationship graph $G = \langle U, E, C\rangle$, the size of smallest clique $k$
Output: set of influential users IUS
1.  CSG $\longleftarrow \varnothing$ ; $O \longleftarrow 0_{|L|\times|L|}$ ; $A \longleftarrow 0_{|L|\times|L|}$
2.  for each user $u_i \in U$ do
3.    if $G_i = \{u_1^i, u_2^i, \cdots, u_k^i\}$ and $G_i \subseteq G$ and $\forall C(u_m^i, u_n^i) \neq 0$ then
4.      $TG \longleftarrow TG \cup G_i$
5.    end if
6.  end for
7.  for $G_i \in TG$ do
8.    $O[i][i] \longleftarrow |G_i|$
9.    for $G_j \in TG$ do
10.     $O[i][j] \longleftarrow |G_i \cap G_j|$
11.   end for
12. end for
13. for $\forall i, j \in L$ and $i \neq j$ do
14.   if $O[i][i] < k$ or $O[i][j] \leq k - 1$ then
15.     $A[i][i] \longleftarrow 0$
16.     $A[i][j] \longleftarrow 0$
17.   else
18.     $A[i][i] \longleftarrow 1$
19.     $A[i][j] \longleftarrow 1$
20.   end if
21. end for
22. for $\forall i, j \in L$ do
23.   if $A[i][j] = 1$ then
24.     CSG $\longleftarrow G_i \cup G_j$
25.   end if
26. end for
27. for each community $cs \in$ CSG do.
28.   for each user $u \in cs$ do
29.     if UNNS$_u >$ UNNS$_{\text{Neighbor}(u)}$ then
30.       IUS $\longleftarrow$ IUS $\cup \{u\}$
31. return IUS
```

Algorithm 2: Extracting influential users.

u_i and user u_j and $|\text{Neighbor}(u_i)|$ represents the number of first-order neighbors of user u_i.

Definition 9 (user nearest neighbor preference similarity, UNNPS). For $\forall u_i \in U$, the user's nearest neighbor preference refers to the average preference similarity between the user and its direct neighbors, which is denoted as UNNPS(u_i) and calculated by

$$\text{UNNPS}(u_i) = \frac{\sum_{u_j \in \text{Neighbor}(u_i)} \text{UPS}(u_i, u_j)}{|\text{Neighbor}(u_i)|}, \qquad (9)$$

where UPS(u_i, u_j) represents the preference similarity of user u_i and user u_j.

Based on the above definition, we extract the initial embedding $X_u = (\text{ULP}_u, \text{UNNRS}_u, \text{UNNPS}_u, \text{UNNS}_u)$ of user u.

3.2.2. Identifying Attack Users Based on the GCN. In previous graph embedding-based group shilling attack detection methods, researchers focused on how to obtain high-quality user node embeddings in the graph. Zhang et al. [19] obtained a low-dimensional embedding vector of nodes in the graph by adopting the Node2vec method that focuses on obtaining the structural characteristics of the user's topological neighborhood but ignores the characteristic information of the nodes themselves. The existing group attack detection methods also use hard classification, in which members from the same group are classified as genuine users or attackers, resulting in the misclassification of some users [18–21]. To this end, we utilize user high-quality embedding features from both implicit and explicit perspectives by combining user initial embeddings with their higher-order topological neighborhood structures based on the GCN and employing influential node labels to identify attack users.

We first extract the high-quality embeddings of user nodes based on the user initial embedding matrix X and the weighted matrix C. The GCN propagation process is formulated as follows:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{C}\tilde{D}^{-1/2}H^{(l)}W^{(l)}\right), \qquad (10)$$

where $H^{(l+1)}$ represents the output after one convolutional layer. $H^{(l)}$ is the input of the lth layer. $H^{(0)} = X_{|N|\times 4}$

represents the user initial embedding matrix. N is the total number of user nodes in the graph, and the feature vector of each user is represented as $X_u = (\mathrm{ULP}_u, \mathrm{UNNRS}_u, \mathrm{UNNPS}_u, \mathrm{UNNS}_u)$. $\tilde{C} = C + I_N$ is the adjacency matrix by adding self-connection. $\tilde{D}$ is the degree matrix, and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. $W^{(l)} \in R^{P \times H}$ denotes the parameter matrix to be trained, P represents the length of the feature matrix, and H represents the number of hidden units. σ is the corresponding activation function, such as $\mathrm{Re\,LU}(\cdot) = \max(0, \cdot)$.

High-quality user embeddings can be obtained after multiple convolutional layers. We utilize two convolutional layers and ReLU as the activation function.

We then calculate the cross-entropy between the real label one-hot vector Y of all influential user nodes and the label vector T predicted by softmax. Subsequently, we utilize the gradient descent method to train the parameter matrix $W^{(0)}$ and $W^{(1)}$. The formula for calculating the loss function is as follows.

$$\mathrm{loss} = -\sum_{l \in \Upsilon_L} \sum_{t=1}^{T} Y_{lf} \ln Z_{lf}, \tag{11}$$

where Υ_L is the set of influential user nodes with labels.

Finally, the resulting model is expressed as

$$Z = f(X, C) = \mathrm{softmax}\left(\hat{C}\mathrm{ReLU}\left(\hat{C}XW^{(0)}\right)W^{(1)}\right), \tag{12}$$

where Z represents the set of user labels after classification by the softmax function. $\hat{C} = \tilde{D}^{-1/2}\tilde{C}\tilde{D}^{-1/2}$ represents the weighted matrix C after symmetric normalization.

The algorithm for detecting attackers is described as follows.

Algorithm 3 is divided into two parts. The first part (lines 1–8) uses GCN semi-supervised classification model training to get the classification result Z of all user nodes. The second part (lines 9–14) filters out the attack users according to the classification result Z.

4. Experimental Evaluation

4.1. Experimental Datasets. The following two datasets are utilized as the experimental datasets to evaluate how well the proposed KC-GCN method performs.

(1) *Netflix dataset (this dataset was constructed to support the participants in the Netflix prize (*http://netflixprize.com*))*: this dataset contains 1,032,938 ratings and the rating time for 17,770 movies by 480,186 users. The ratings are expressed in integers from 1 to 5, where 1 and 5 indicate disliked and most liked, respectively. We randomly sample 215,884 ratings and the rating time of 2000 users on 4000 movies for use in the experimental dataset. Similar to the previous research, the 2000 extracted users are regarded as genuine users. Multiple group attack profiles are generated and injected into the dataset by using the group shilling attack model introduced in Section 2.1. Under $\mathrm{GSAGen_l}$ Ave, $\mathrm{GSAGen_l}$ Ran, and $\mathrm{GSAGen_l}$ AOP, 10 attack groups are generated each time. The filler size is set to 2%, and the attack size is set to 2.5%, 5%, 7.5%, and 10%. The target items in each attack group are randomly selected from unpopular items. Two target item strategies are set (denoted as ST1 and ST2) to prove the influence of the relationship between the attack users in the same group on the detection performance. ST1 means that all attackers of the same group rate all the target items (number of target items in the experiments: 3). ST2 means that each attacker of the same group rate any three of the five target items. This results to $4 * 2 * 3 * 2 = 48$ experimental datasets generated. We generate loose group attack profiles to verify the universality of the proposed method using the $\mathrm{GSAGen_l}$ GOAT and $\mathrm{GSAGen_l}$ Mixed attack models introduced in Section 2.1 and two target item strategies. The dataset generated based on the $\mathrm{GSAGen_l}$ GOAT attack model specifically contains eight attack groups. The dataset generated based on the $\mathrm{GSAGen_l}$ Mixed attack model contains 26 attack groups. For convenience of description, under the condition of the target item strategies ST1 and ST2, the shilling attack groups generated are denoted as loosely and tightly coupled shilling groups, respectively

(2) *Amazon dataset [36]*: this dataset contains 1,205,125 ratings and the rating time on 136,785 products from 645,072 users crawled from Amazon.cn until August 20, 2012. The ratings are integers between 1 and 5, which indicate disliked and most liked, respectively. We evaluate the proposed method using a sampled dataset with 5055 labeled users. The dataset consists of 53,777 ratings of 17,610 products by 3118 genuine users and 1937 attack users

4.2. Evaluation Metrics. Three metrics including precision, recall, and F1-measure are used to evaluate the detection performance of the KC-GCN:

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \tag{13}$$

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{14}$$

$$\text{F1-measure} = \frac{2 \times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}, \tag{15}$$

where TP represents the number of attackers accurately recognized, FN represents the number of attackers mistaken for genuine users, and FP represents the number of genuine users mistaken for attackers.

4.3. Parameter Selection. Figure 2 shows how the F1-measure of the KC-GCN is influenced by parameters θ and k on the Netflix and Amazon datasets. In Figure 2(a), the F1-measure of KC-GCN is the highest for detecting the

```
Input: the weighted user's relationship graph $G$, the influential user's set IUS, the user initial embedding matrix $X$, and maximum training epoch $K$
Output: set of attack users AU
1.  AU $\longleftarrow \varnothing$
2.  for $k = 1$ to $K$ do
3.      compute $Z$ according to Eq. (12)
4.      compute loss according to Eq. (11)
5.      Gradient zeroing
6.      Back propagation calculation gradient value
7.      Update parameters by using gradient descent
8.  end for
9.  for each $z \in Z$ do
10.         if $z = 1$ then
11.             AU $\longleftarrow$ AU $\cup \{z\}$
12.         end if
13.  end for
14.  return AU
```

Algorithm 3: Detecting attack users.

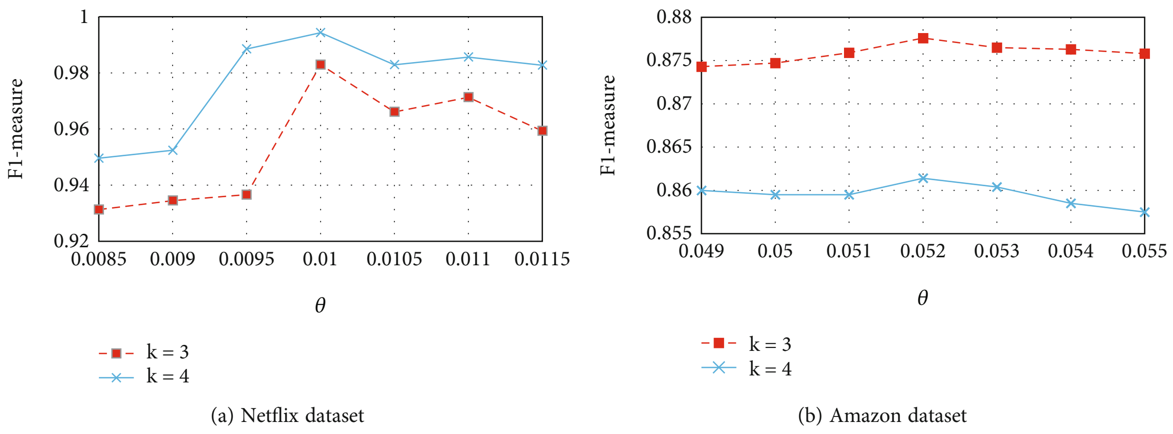


Figure 2: The influence of parameters θ and k on the F1-measure of KC-GCN.

GSAGen$_l$ Ran attack on the Netflix dataset under a θ value set to 0.01. Under a smaller θ value, the user relationship graph contains a large number of weak relationship edges, and the community structure is not obvious, leading to a decrease of the detection precision. At a larger θ, the user relationship graph shows an obvious community structure, but some user nodes are filtered from the graph, thereby degrading the detection recall. Moreover, $k = 4$ has a superior detection performance than $k = 3$; therefore, for the Netflix dataset, we set k to 4 and θ to 0.01. Figure 2(b) shows that when $\theta = 0.052$ and $k = 3$, the F1-measure of KC-GCN is close to 0.8776 on the sampled Amazon dataset, which is the best. Therefore, we set k to 3 and θ to 0.052 for the sampled Amazon dataset.

4.4. Experimental Results and Analysis. To verify the effectiveness of KC-GCN, we compare the precision, recall, and F1-measure of KC-GCN with the following methods.

We assess the precision, recall, and F1-measure of KC-GCN in comparison to the following methods to confirm its efficacy.

(1) *Catch the Black Sheep (CBS) [27]*: this detection method uses label propagation to iteratively calculate the malicious probability of users and items, which needs the number of spammers and a certain number of seed users in advance. In contrast to the experiments, the number of seed users on the two datasets is consistent with that of our method

(2) *GAGE [19]*: this is an unsupervised group shilling attack detection method based on graph embedding, which learns the low-dimensional vector representation of nodes in the user relationship graph using Node2vec and obtains attack groups through clustering. In the experiments, the working strategy is adjusted by setting parameters $p = 7$ and $q = 0.2$ and group size (GS) = 30

(3) *TP-GBF [20]*: this is an unsupervised group shilling attack detection method based on the strong association between the group members and the group behavior features, which uses a topological potential-based community partition algorithm to

generate tight subgraphs as candidate groups and cluster attack groups by group behavior features. In the experiments, parameter θ is set to 2, while parameter σ is set to 1 and 0.47 in the Netflix and Amazon datasets, respectively

4.4.1. Comparison of the Detection Results on the Netflix Dataset. Table 3 compares KC-GCN and three baseline methods to identify the group shilling attacks with tightly coupled shilling groups at various attack sizes on the Netflix dataset. In Table 3, the precision and recall values of the CBS remain stable for detecting three types of group shilling attacks, only slightly changing the attack size from 2.5% to 10%. The CBS detection performance is much lower than that of KC-GCN when detecting various types of group shilling attacks with tightly coupled shilling groups, albeit the number of attackers is assumed in advance. Meanwhile, the precision values of GAGE under three types of group attacks are the worst, indicating the misclassification of a large number of genuine users as attack ones. This happened because GAGE generates the user node feature vectors using Node2-vec, from which a certain degree of randomness may cause some genuine and attack users to be divided into the same candidate group. The detection performance of TP-GBF is better than those of CBS and GAGE when detecting the group shilling attacks with tightly coupled shilling groups. The detection recall was not high under the GSAGen$_l$ AOP. Compared with CBS, GAGE, and TP-GBF, KC-GCN shows the best detection performance because it can extract more effective features when correctly differentiating attack profiles from genuine ones. KC-GCN uses a weighted graph to aggregate the neighbor features, thereby effectively avoiding the merging of user features with different labels. It can fully integrate the user node and structural features, further increasing the difference between attackers and normal users. In conclusion, KC-GCN outperforms the baselines for detecting various types of group shilling attacks with tightly coupled shilling groups at various attack sizes on the Netflix dataset.

Table 4 compares the performances of our proposed KC-GCN and three baseline methods in terms of detecting group shilling attacks with loosely coupled shilling groups at various attack sizes on the Netflix dataset. In Table 4, the precision and recall values of CBS under the three attack models significantly decrease when the relationship between users within the attack group is weakened. This indicates that improving the detection performance is difficult when relying only on the rating bias. The GAGE performance becomes better with the attack size increase, but its precision greatly fluctuates because it may falsely identify some normal users as attackers. TP-GBF shows an excellent detection performance under the GSAGen$_l$ Ran and GSAGen$_l$ Ave attacks but is less effective under the GSAGen$_l$ AOP attack. Its detection performance becomes extremely unstable with the change of the attack size. KC-GCN yields the best detection performance among the four methods. It shows a slight decline in detecting loosely coupled shilling groups mainly because the feature differences between the attackers and the genuine users are weakened with a looser relationship in a group. In conclusion, KC-GCN outperforms the baseline methods in detecting various types of group shilling attacks with loosely coupled shilling groups at various attack sizes on the Netflix dataset.

Figure 3 compares the detection results of the four detection methods under the GSAGen$_l$ GOAT attack on the Netflix dataset. In the Netflix dataset, the precision, recall, and F1-measure of CBS when identifying tightly and loosely coupled shilling groups are 0.6791, 0.8184, and 0.7422 and 0.6352, 0.7656, and 0.6943, respectively. The detection performance of CBS is constrained by the number and influence of seed users. These results also indicate that CBS can achieve superior detection performance when a closer relationship exists between the group members. The precision, recall, and F1-measure of GAGE for detecting the tightly and loosely coupled shilling groups are 0.4046, 0.6968, and 0.5119 and 0.3157, 0.9120, and 0.4690, respectively. These results indicate that GAGE is less effective on the Netflix dataset under the GSAGen$_l$ GOAT attack because the GOAT attack model uses the genuine user profile as a template to generate the attack profile, which is highly similar to the genuine user. However, the user node feature vector obtained by the Node2vec method cannot effectively distinguish genuine users and attackers. For TP-GBF, the precision, recall, and F1-measure of the tightly coupled shilling groups are 0.9905, 0.7269, and 0.8385, respectively, while those for the loosely coupled shilling groups are 0.7036, 0.5000, and 0.5846, respectively. TP-GBF shows an extremely high precision in identifying the tightly coupled shilling groups; nevertheless, the recall of TP-GBF is poor when identifying the loosely coupled shilling groups because it cannot distinguish weakly related attack groups. Figure 3 shows that GAGE and TP-GBF have poor performances when detecting attack groups generated based on GOAT because the profiles generated by GOAT are very similar to the genuine profiles. The precision, recall, and F1-measure of KC-GCN for detecting the tightly coupled shilling groups are 1, 0.9857, and 0.9928, respectively, while those for the loosely coupled shilling groups are 1, 0.9282, and 0.9628, respectively. These results show that KC-GCN is effective and outperforms the three baseline methods for detecting groups under the GSAGen$_l$ GOAT attack on the Netflix dataset. In other words, the feature differences between the attackers and the genuine users can be reinforced by using the weighted GCN to aggregate the user node features.

Figure 4 compares the detection results of the four detection methods under the GSAGen$_l$ Mixed attack on the Netflix dataset. In this dataset, the precision, recall, and F1-measure of CBS for identifying the tightly and loosely coupled shilling groups are 0.8190, 0.9992, and 0.9002 and 0.8191, 0.9996, and 0.9004, respectively. CBS remains stable when detecting the tightly and loosely coupled group shilling attacks. GAGE shows precision, recall, and F1-measure of 0.9542, 0.9275, and 0.9407, respectively, for the tightly coupled shilling groups. For the loosely coupled shilling groups, the precision, recall, and F1-measure of GAGE are 0.8212, 0.9376, and 0.8759, respectively. Its detection performance significantly declines with the weakening user relationships. The main reason for this is that with the

Table 3: Comparison between KC-GCN and other detection methods for detecting group shilling attacks with tightly coupled shilling groups at various attack sizes on the Netflix dataset.

Attack type	Metrics	Method	Attack size			
			2.5%	5%	7.5%	10%
$GSAGen_l$ Ran	Precision	CBS	0.7811	0.7979	0.7974	0.8019
		GAGE	0.5630	0.8906	0.7856	0.9124
		TP-GBF	0.9944	1.0000	0.9987	0.9973
		KC-GCN	0.9954	0.9937	0.9965	1.0000
	Recall	CBS	0.9716	0.9879	0.9870	0.9912
		GAGE	0.9918	0.9904	0.9815	0.9696
		TP-GBF	0.9152	0.9428	0.9054	0.8189
		KC-GCN	0.9487	0.9822	0.9725	0.9886
	F1-measure	CBS	0.8660	0.8828	0.8821	0.8866
		GAGE	0.7183	0.9379	0.8727	0.9401
		TP-GBF	0.9532	0.9706	0.9498	0.8993
		KC-GCN	0.9715	0.9879	0.9844	0.9943
$GSAGen_l$ Ave	Precision	CBS	0.8045	0.8076	0.8125	0.8098
		GAGE	0.8584	0.7445	0.8913	0.7624
		TP-GBF	0.9944	0.9870	1.0000	0.9989
		KC-GCN	0.9925	0.9932	0.9918	0.9986
	Recall	CBS	0.9797	0.9815	0.9865	0.9838
		GAGE	0.9876	0.9856	0.8518	0.9748
		TP-GBF	0.9383	0.9159	0.8941	0.9782
		KC-GCN	0.9500	0.9735	0.9871	0.9914
	F1-measure	CBS	0.8835	0.8861	0.8911	0.8884
		GAGE	0.9185	0.8483	0.8711	0.8556
		TP-GBF	0.9655	0.9501	0.9441	0.9884
		KC-GCN	0.9708	0.9833	0.9894	0.9950
$GSAGen_l$ AOP	Precision	CBS	0.6915	0.7035	0.7242	0.7214
		GAGE	0.7051	0.7479	0.806	0.7507
		TP-GBF	0.9764	0.9837	0.9854	0.9914
		KC-GCN	0.9740	0.9875	0.9773	0.9886
	Recall	CBS	0.8512	0.8657	0.8866	0.8843
		GAGE	0.9806	0.9322	0.9725	0.9679
		TP-GBF	0.8118	0.8025	0.8477	0.7744
		KC-GCN	0.9615	0.9080	0.9556	0.9255
	F1-measure	CBS	0.7631	0.7762	0.7972	0.7946
		GAGE	0.8203	0.8299	0.8815	0.8456
		TP-GBF	0.8865	0.8839	0.9114	0.8696
		KC-GCN	0.9677	0.9461	0.9663	0.9560

weakening user relationship in the group, its spatial structure changes, resulting in obvious changes in the initial user embedding and a significant decrease in the detection performance. The precision, recall, and F1-measure of TP-GBF for detecting tightly coupled shilling groups are 0.7209, 0.8204, and 0.7674, respectively, while those for loosely coupled shilling groups are 0.6620, 0.6141, and 0.6372, respectively. TP-GBF is less effective on the Netflix dataset under the $GSAGen_l$ Mixed attack. The precision, recall, and F1-measure of KC-GCN for identifying the tightly and loosely coupled shilling groups are 0.9978, 0.9430, and 0.9696 and 0.9583, 0.9705, and 0.9644, respectively. These findings demonstrate that KC-GCN is effective and outperforms the three baseline methods in terms of precision and F1-measure under the $GSAGen_l$ Mixed attack on the Netflix dataset.

Figure 5 shows the results of the four detection methods on the sampled Amazon dataset. The detection performance of KC-GCN is superior to that of the baseline methods on this dataset, yielding precision, recall, and F1-measure of

Table 4: Comparison between KC-GCN and other detection methods for detecting group shilling attacks with loosely coupled shilling groups at various attack sizes on the Netflix dataset.

Attack type	Metrics	Method	Attack size 2.5%	5%	7.5%	10%
GSAGen$_l$ Ran	Precision	CBS	0.7993	0.6441	0.6422	0.6479
		GAGE	0.8749	0.8102	0.8749	0.9078
		TP-GBF	1.0000	0.9985	0.8247	0.6856
		KC-GCN	0.9683	0.9852	0.9858	0.9593
	Recall	CBS	0.9730	0.7835	0.7796	0.7866
		GAGE	0.9459	0.9868	0.9509	0.9933
		TP-GBF	0.8555	0.8242	0.8451	0.7768
		KC-GCN	0.9313	0.8965	0.8910	0.9940
	F1-measure	CBS	0.8777	0.7070	0.7042	0.7105
		GAGE	0.9090	0.8898	0.9113	0.9486
		TP-GBF	0.9221	0.9030	0.8348	0.7284
		KC-GCN	0.9494	0.9399	0.9360	0.9763
GSAGen$_l$ Ave	Precision	CBS	0.7972	0.6439	0.6439	0.6487
		GAGE	0.6345	0.8544	0.8491	0.9070
		TP-GBF	0.4497	0.5362	0.5939	0.9137
		KC-GCN	0.9542	0.9783	0.9800	0.9600
	Recall	CBS	0.9723	0.7823	0.7823	0.7879
		GAGE	0.9884	0.9482	0.9900	0.9138
		TP-GBF	0.7055	0.7977	0.7687	0.7704
		KC-GCN	0.9843	0.9184	0.9188	0.9941
	F1-measure	CBS	0.8761	0.7064	0.7064	0.7115
		GAGE	0.7729	0.8988	0.9142	0.9104
		TP-GBF	0.5493	0.6413	0.6701	0.8360
		KC-GCN	0.9690	0.9474	0.9484	0.9767
GSAGen$_l$ AOP	Precision	CBS	0.5401	0.5624	0.5655	0.5569
		GAGE	0.5501	0.6043	0.4545	0.7517
		TP-GBF	0.8733	0.5643	0.7359	0.6868
		KC-GCN	0.9589	0.9294	0.9620	0.9540
	Recall	CBS	0.6664	0.6906	0.6944	0.6824
		GAGE	0.9151	0.9170	0.9444	0.9372
		TP-GBF	0.7773	0.7208	0.7655	0.7361
		KC-GCN	0.9211	0.9875	0.9048	0.9432
	F1-measure	CBS	0.5967	0.6200	0.6233	0.6133
		GAGE	0.6871	0.7285	0.6134	0.8343
		TP-GBF	0.8225	0.6330	0.7504	0.7106
		KC-GCN	0.9396	0.9576	0.9325	0.9486

0.9179, 0.8407, and 0.8776, respectively. This indicates that KC-GCN can effectively combine user node and graph structure features to construct new user features by using GCN, which can distinguish genuine and attack users on the sampled Amazon dataset. The precision, recall, and F1-measure of CBS are 0.6836, 0.8323, and 0.7507, respectively. This means that CBS can detect attack users on the Amazon dataset but that its detection performance is determined by the number of seed users. Meanwhile, GAGE exhibits 0.8004, 0.9277, and 0.8594 of precision, recall, and F1-measure, respectively. The result indicates that GAGE has a certain randomness when sampling with Node2vec, which leads to a bias in the division of the candidate groups, and a precision measurement performance is lower than that of KC-GCN. The precision, recall, and F1-measure of TP-GBF are 0.9283, 0.6467, and 0.7623, respectively. This precision is not much higher than that of KC-GCN, but its recall is lower than that of KC-GCN, indicating that TP-GBF may have filtered out some attack groups with a low density. In summary, KC-GCN shows a superior detection performance over GAGE, CBS, and TP-GBF on the sampled Amazon dataset.

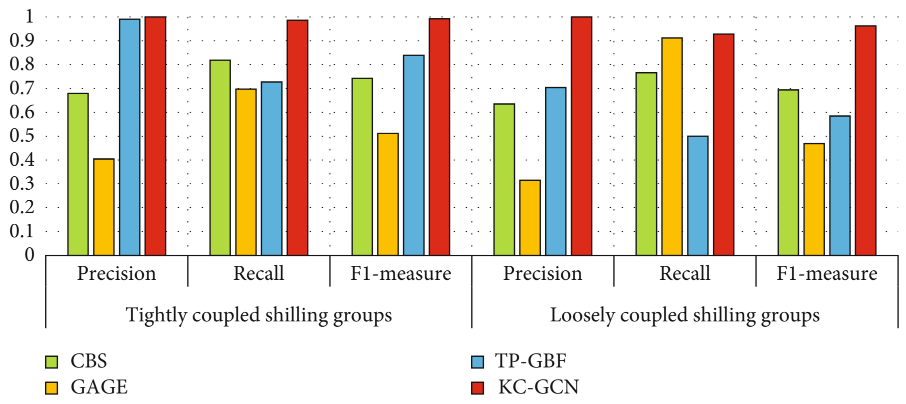


Figure 3: Comparison of the detection results of the four detection methods on the Netflix dataset under the GSAGen$_l$ GOAT attack.

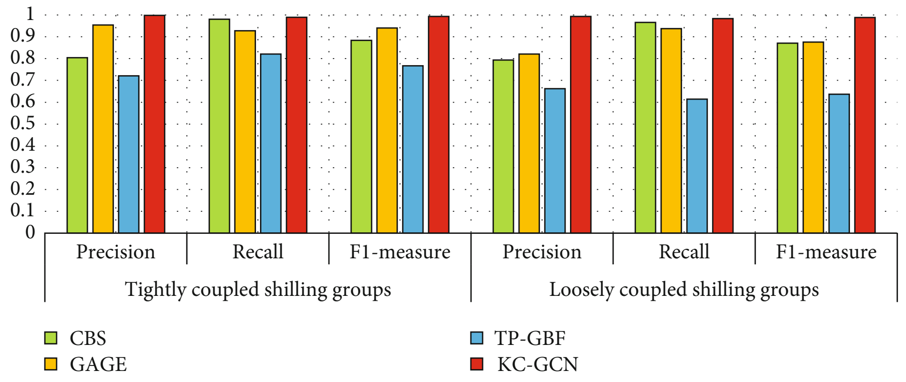


Figure 4: Comparison of the detection results of the four detection methods on the Netflix dataset under the GSAGen$_l$ Mixed attack.

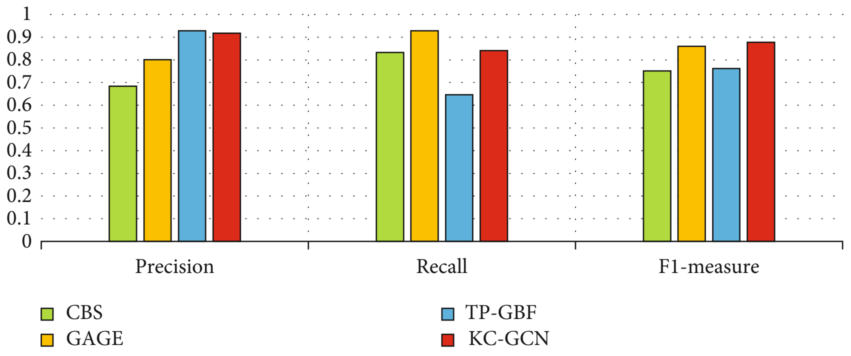


Figure 5: Comparison of the detection results of the four detection methods on the Amazon dataset.

5. Conclusions and Future Work

In this work, we put forward a two-stage semi-supervised model to validly detect various types of group shilling attacks on recommender systems. First, we construct a user relationship graph and spot the influential users. In the graph, the edge weight is calculated by analyzing the user similarity over suspicious time intervals on each item. Next, we generate the initial user embeddings based on the proposed four indicators describing the behavior difference between attack and genuine users. A GCN-based classifier is trained, and the attack users are detected based on the influential user labels. The experimental results prove the effectiveness and the generality of KC-GCN.

In the future work, we will automatically determine the labels of most influential users by further analyzing the structural properties of the weighted user relationship graph. We will also study the multiaspect data [37] to further help identify users of group shilling attack.

Data Availability

The data used to support the findings of this study are available from the second author (renjich@163.com) upon request.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Science and Technology Project of the Hebei Education Department (ZD2022105), the Natural Science Foundation of Hebei Province, China (F2020201023), and the High-Level Personnel Starting Project of Hebei University (521100221089).

References

[1] H. Li, K. Wang, Y. Sun, and X. Mou, "Application of recommendation systems based on deep learning," in *Recent Challenges in Intelligent Information and Database Systems. ACIIDS 2021*, Communications in Computer and Information Science, T. P. Hong, K. Wojtkiewicz, R. Chawuthai, and P. Sitek, Eds., pp. 85–97, Springer, Singapore, 2021.

[2] Q. Shambour, "A deep learning based algorithm for multi-criteria recommender systems," *Knowledge-Based Systems*, vol. 211, article 106545, 2021.

[3] N. Nassar, A. Jafar, and Y. Rahhal, "A novel deep multi-criteria collaborative filtering model for recommendation system," *Knowledge-Based Systems*, vol. 187, no. 7, pp. 104811.1–104811.7, 2020.

[4] Y. Feng, F. Lv, W. Shen et al., "Deep session interest network for click-through rate prediction," 2019, https://arxiv.org/abs/1905.06482.

[5] F. Lv, T. Jin, C. Yu et al., "SDM: sequential deep matching model for online large-scale recommender system," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 2635–2643, Beijing, China, 2019.

[6] S. Zhang, H. Liu, J. He, S. Han, and X. Du, "Deep sequential model for anchor recommendation on live streaming platforms," *Big Data Mining and Analytics*, vol. 4, no. 3, pp. 173–182, 2021.

[7] P. Nitu, J. Coelho, and P. Madiraju, "Improvising personalized travel recommendation system with recency effects," *Big Data Mining and Analytics*, vol. 4, no. 3, pp. 139–154, 2021.

[8] T. Li, C. Li, J. Luo, and L. Song, "Wireless recommendations for internet of vehicles: recent advances, challenges, and opportunities," *Intelligent and Converged Networks*, vol. 1, no. 1, pp. 1–17, 2020.

[9] H. Li, M. Gao, F. Zhou, Y. Wang, Q. Fan, and L. Yang, "Fusing hypergraph spectral features for shilling attack detection," *Journal of Information Security and Applications*, vol. 63, article 103051, 2021.

[10] N. Hurley, Z. Cheng, and M. Zhang, "Statistical attack detection," in *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pp. 149–156, New York, USA, 2009.

[11] F. Wu, M. Gao, J. Yu, Z. Wang, K. Liu, and X. Wang, "Ready for emerging threats to recommender systems? A graph convolution-based generative shilling attack," *Information Sciences*, vol. 578, pp. 683–701, 2021.

[12] X. Su, H. Zeng, and Z. Chen, "Finding group shilling in recommender system," in *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pp. 960-961, China, 2005.

[13] L. Yang and X. Niu, "A genre trust model for defending shilling attacks in recommender systems," *Complex & Intelligent Systems*, pp. 1–14, 2021.

[14] S. Rani, M. Kaur, M. Kumar, V. Ravi, U. Ghosh, and J. R. Mohanty, "Detection of shilling attack in recommender system for YouTube video statistics using machine learning techniques," *Soft Computing*, vol. 27, pp. 377–389, 2023.

[15] Y. Wang, Z. Wu, Z. Bu, J. Cao, and D. Yang, "Discovering shilling groups in a real e-commerce platform," *Online Information Review*, vol. 40, no. 1, pp. 62–78, 2016.

[16] W. Zhou, Y. S. Koh, J. Wen, S. Alam, and G. Dobbie, "Detection of abnormal profiles on group attacks in recommender systems," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, Gold Coast Queensland, Australia, 2014.

[17] W. Zhou, J. Wen, Q. Xiong, M. Gao, and J. Zeng, "SVM-TIA a shilling attack detection method based on SVM and target item analysis in recommender systems," *Neurocomputing*, vol. 210, no. 19, pp. 197–205, 2016.

[18] F. Zhang and S. Wang, "Detecting group shilling attacks in online recommender systems based on bisecting *K*-means clustering," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 5, pp. 1189–1199, 2020.

[19] F. Zhang, Y. Qu, Y. Xu, and S. Wang, "Graph embedding-based approach for detecting group shilling attacks in collaborative recommender systems," *Knowledge-Based Systems*, vol. 199, article 105984, 2020.

[20] H. Yu, H. Zheng, Y. Xu, R. Ma, D. Gao, and F. Zhang, "Detecting group shilling attacks in recommender systems based on maximum dense subtensor mining," in *2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, Melbourne, Australia, 2021.

[21] H. Cai and F. Zhang, "An unsupervised approach for detecting group shilling attacks in recommender systems based on topological potential and group behaviour features," *Security and Communication Networks*, vol. 2021, Article ID 2907691, 18 pages, 2021.

[22] Y. Gao, X. Yu, and H. Zhang, "Uncovering overlapping community structure in static and dynamic networks," *Knowledge-Based Systems*, vol. 201-202, p. 106060, 2020.

[23] J. Zhou, G. Cui, S. Hu et al., "Graph neural networks: a review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[24] Y. Wang, Z. Wu, J. Cao, and C. Fang, "Towards a tricksy group shilling attack model against recommender systems," in *Advanced Data Mining and Applications. ADMA 2012*, S. Zhou, S. Zhang, and G. Karypis, Eds., vol. 7713 of Lecture Notes in Computer Science, pp. 675–688, Springer, Berlin, Heidelberg, 2012.

[25] P. A. Chirita, W. Nejdl, and C. Zamfir, "Preventing shilling attacks in online recommender systems," in *Proceedings of the 7th annual ACM international workshop on Web information and data management*, pp. 67–74, New York, USA, 2005.

[26] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik, "Detecting profile injection attacks in collaborative recommender systems," in *The 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06)*, p. 23, San Francisco, USA, 2006.

[27] Y. Zhang, Y. Tan, M. Zhang, and Y. Liu, "Catch the black sheep: unified framework for shilling attack detection based on fraudulent action propagation," in *Proceedings of the 24th International Conference on Artificial Intelligence*, pp. 2408–2414, Bunos Aires, Argentina, 2015.

[28] F. Zhang, Z. Zhang, P. Zhang, and S. Wang, "UD-HMM: an unsupervised method for shilling attack detection based on hidden Markov model and hierarchical clustering," *Knowledge-Based Systems*, vol. 148, pp. 146–166, 2018.

[29] Z. Yang, Q. Sun, and Y. Zhang, "Probabilistic inference and trustworthiness evaluation of associative links toward malicious attack detection for online recommendations," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, pp. 1–896, 2020.

[30] Y. Hao and F. Zhang, "An unsupervised detection method for shilling attacks based on deep learning and community detection," *Soft Computing*, vol. 25, no. 1, pp. 477–494, 2021.

[31] M. Ebrahimian and R. Kashef, "Detecting shilling attacks using hybrid deep learning models," *Symmetry*, vol. 12, no. 11, pp. 1805–1821, 2020.

[32] Q. Zhou and L. Duan, "Semi-supervised recommendation attack detection based on co-forest," *Computers & Security*, vol. 109, article 102390, 2021.

[33] S. Zhang, H. Yin, and T. Chen, "GCN-based user representation learning for unifying robust recommendation and fraudster detection," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 689–698, Xi'an, China, 2020.

[34] Y. Li, J. Fan, Y. Wang, and K. L. Tan, "Influence maximization on social graphs: a survey," *IEEE Transactions on Knowledge & Data Engineering*, vol. 30, no. 10, pp. 1852–1872, 2018.

[35] X. Liu, S. Wu, C. Liu, and Y. Zhang, "Social network node influence maximization method combined with degree discount and local node optimization," *Social Network Analysis and Mining*, vol. 11, no. 1, article 31, 2021.

[36] C. Xu, J. Zhang, and C. Long, "Uncovering collusive spammers in Chinese review websites," in *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management - CIKM '13*, pp. 979–988, Burlingame, USA, 2013.

[37] L. Qi, Y. Yang, X. Zhou, W. Rafique, and J. Ma, "Fast anomaly identification based on multiaspect data streams for intelligent intrusion detection toward secure industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 9, pp. 6503–6511, 2022.

Hindawi
Wireless Communications and Mobile Computing
Volume 2023, Article ID 9212269, 12 pages
https://doi.org/10.1155/2023/9212269

Research Article

A Smart Contract Vulnerability Detection Model Based on Syntactic and Semantic Fusion Learning

Daojun Han,[1] **Qiuyue Li**,[2] **Lei Zhang**,[1] **and Tao Xu**[1]

[1]*Henan Engineering Research Center of Intelligent Technology and Application, Henan University, Kaifeng 475004, China*
[2]*School of Computer and Information Engineering, Henan University, Kaifeng 475000, China*

Correspondence should be addressed to Lei Zhang; zhanglei@henu.edu.cn

Received 7 July 2022; Revised 4 November 2022; Accepted 24 November 2022; Published 3 February 2023

Academic Editor: Changyan Yi

As a trusted decentralized application, smart contracts manage a large number of digital assets on the blockchain. Vulnerability detection of smart contracts is an important part of ensuring the security of digital assets. At present, many researchers extract features of smart contract source code for vulnerability detection based on deep learning methods. However, the current research mainly focuses on the single representation form of the source code, which cannot fully obtain the rich semantic and structural information contained in the source code, so it is not conducive to the detection of various and complex smart contract vulnerabilities. Aiming at this problem, this paper proposes a vulnerability detection model based on the fusion of syntax and semantic features. The syntactic and semantic representation of the source code is obtained from the abstract syntax tree and control flow graph of the smart contract through TextCNN and Graph Neural Network. The syntactic and semantic features are fused, and the fused features are used to detect vulnerabilities. Experiments show that the detection accuracy and recall rate of this model have been improved on the detection tasks of five types of vulnerabilities, with an average precision of 96% and a recall rate of 90%, which can effectively identify smart contract vulnerabilities.

1. Introduction

Smart contracts were proposed by Nick Szabo in 1995 and are defined as " a set of promises, specified in digital form, including protocols within which the parties perform on these promises." [1]. A smart contract defines a contract in digital form, and the contract is automatically executed when the contract participants meet the conditions required by the smart contract. Due to the lack of a trusted execution environment, smart contracts were not really implemented before blockchain technology was proposed. In 2008, Satoshi Nakamoto proposed the concept of blockchain in "Bitcoin: A Peer-to-Peer Electronic Cash System", which provides a trusted and immutable execution environment for smart contracts. In 2014, Vitalik Buterin proposed Ethereum, inspired by Bitcoin. Ethereum is the first blockchain platform to support developers in creating smart contracts and decentralized applications. Through smart contracts, developers can create decentralized applications on platforms such as Ethereum to achieve trusted transactions. Compared with traditional contracts, smart contracts can improve transaction efficiency and have the characteristics of decentralization and immutability. Smart contracts provide security methods superior to traditional contracts, but like other programming languages, smart contracts also have some security vulnerabilities. In the 2016, the DAO incident, attackers stole more than 3.6 million ethers by exploiting a vulnerability in the smart contract code. In 2018, there was a major vulnerability in the BEC smart contract. The attacker used the integer overflow problem of the transfer function to generate tokens indefinitely, causing a loss of about 6 billion [2]. Smart contracts involve the digital properties of a large number of users, and loopholes in smart contracts will cause the loss of digital assets. At the same time, the immutable property of the blockchain makes it difficult to repair the contract vulnerability once it occurs, so it is necessary to check the contract vulnerability before deploying the smart contract on the chain.

According to whether the code is executed during detection, the early vulnerability detection methods are mainly static analysis and dynamic analysis methods [3–6]. Static analysis methods can analyze the complete control flow and data flow of the code and have a high level of code coverage [7, 8]. The dynamic analysis tool executes the target program on a real system or in an emulator that can accurately detect program errors [9, 10]. However, using static analysis and dynamic analysis methods to detect smart contracts need to rely on artificially defined expert rules, which results in low detection efficiency and a long time. In recent years, researchers have begun to use deep learning methods for vulnerability detection. Using deep learning, effective features can be automatically extracted from source code without relying on expert-defined vulnerability features. But, there is a diversity of features in the source code, and when applying deep learning methods for vulnerability detection, it is necessary to consider how to better extract features from the source code of smart contracts. Most current research has focused on single feature representations, with some models using program source code or binary files directly as input to extract key feature information such as identifiers, function types, and operators associated with character streams [11–14]; or converting source code into an abstract syntax tree from which the syntactic features of the source code are extracted [15–18]. The source code of smart contracts contains rich feature information, and a single code representation cannot retain the rich syntactic and semantic information in the source code. When using deep learning methods, it is necessary to consider building an appropriate data representation method to extract the feature information related to vulnerabilities to the greatest extent.

This paper proposes a multifeature fusion vulnerability detection model based on deep learning in order to solve the problem of insufficient code representation ability in the current smart contract vulnerability detection model. The model can better extract the source code features, enhance the code representation ability, and improve the accuracy of smart contract vulnerability detection. The model obtains the syntactic and semantic features of smart contracts through Abstract Syntax Tree (AST) and Control Flow Graph (CFG), and constructs a feature space through syntactic and semantic fusion vectors. The establishment of the model is mainly divided into three steps: First, the Ethereum smart contracts are collected, followed by balancing the training dataset using a weighted random sampling method. Then, the AST and CFG of the smart contract are obtained from the Solidity source code, the static features of the contract are extracted from the AST and CFG. Finally, the syntactic and semantic features are concatenated to obtain the feature fusion vector, as well as to use the fusion vector for vulnerability detection of smart contracts. Fusion features can better characterize the vulnerability-related features in the source code, from which the model learns vulnerability patterns for effective and fast detection of vulnerabilities.

The rest of this article is organized as follows. Section 2 discusses related work. Section 3 introduces the types of smart contract vulnerabilities. Section 4 presents the general framework and detailed steps of the model. Section 5 shows the results of our experiments, evaluating the accuracy, precision, recall, and F1 values of the model. Finally, we conclude this paper in Section 6.

```
contract OverflowLoop{
  uint256 public count;

  function OFloop(uint256[] _array) public {
      count = 0;
      for(uint8 i = 0; i < _array.length; i++){
          count++;
      }
  }
}
```

Figure 1: Integer overflow vulnerability example.

2. Background

In this section, several types of smart contract vulnerabilities studied in this paper are presented. The vulnerabilities studied in this paper can be divided into two layers: the Solidity code layer and the blockchain system layer. There are many types of vulnerabilities in the Solidity code layer, including reentrancy vulnerability, insecure arithmetic, permission control vulnerability, denial of service vulnerability, and unknown function call vulnerability. The blockchain system layer includes Timestamp Dependency vulnerability, Block Parameter Dependency vulnerability, and Transaction-Ordering Dependence vulnerability [19–21], among others. This paper mainly studies insecure arithmetic, reentrancy, timestamp dependency, and implicit visibility vulnerabilities that are harmful to the blockchain.

2.1. Insecure Arithmetic. Integer overflow and integer underflow vulnerabilities are caused by values that are outside or below the defined range of the integer type. In computer languages, integer type numbers have maximum and minimum values. In blockchain, integers are unsigned numbers in the range of 0 to 255 [22]. If the maximum value of 255 is exceeded, it will overflow and cause a zero return situation. If the value is 0, subtracting 1 will cause the underflow to become the maximum value. The addition and multiplication operations of numbers can cause overflow problems, and the subtraction of numbers can create underflow problems. Figure 1 contains a function that has the risk of integer overflow. If an attacker passes a value of length greater than 255 to the OFloop () function, an integer overflow will occur for uint8 i. The loop condition $i <$ _array.length is always satisfied, and the loop will keep executing until all the gas fee is consumed.

2.2. Reentrancy. The reentrancy vulnerability is a serious vulnerability. Ethereum smart contracts are able to call and use code from other external contracts to send ether to various external user addresses. The operation of calling an external contract or sending ether to an address requires the contract to submit external calls. An attacker can hijack these external calls to force the contract to execute further code, including calls to itself. If the transfer function uses the call. Value () function to transfer money, the call. Value () function will automatically trigger the fallback () function. If the transfer function modifies the status variable of the

```
1 contract EtherStore {
2
3    uint256 public withdrawalLimit = 1 ether;
4    mapping(address => uint256) public lastWithdrawTime;
5    mapping(address => uint256) public balances;
6
7    function depositFunds() public payable {
8        balances[msg.sender] += msg.value;
9    }
10
11   function withdrawFunds (uint256 _weiToWithdraw) public {
12       require(balances[msg.sender] >= _weiToWithdraw);
13       require(_weiToWithdraw <= withdrawalLimit);
14       require(now >= lastWithdrawTime[msg.sender] + 1 weeks);
15       require(msg.sender.call.value(_weiToWithdraw)());
16       balances[msg.sender] -= _weiToWithdraw;
17       lastWithdrawTime[msg.sender] = now;
18   }
19
20 }
```

```
1 contract Attack {
2
3  EtherStore public etherStore;
4  constructor(address _etherStoreAddress) {
5      etherStore = EtherStore(_etherStoreAddress);
6  }
7  function pwnEtherStore() public payable {
8      require(msg.value >= 1 ether);
9      etherStore.depositFunds.value(1 ether) ();
10      etherStore.withdrawFunds(1 ether);
11  }
12  function collectEther() public {
13      msg.sender.transfer(this.balance);
14  }
15  function () payable {
16      if (etherStore.balance > 1 ether) {
17          etherStore.withdrawFunds(1 ether);
18      }
19  }
20 }
```

Figure 2: Reentrancy vulnerability example.

```
1 contract Roulette {
2     uint public pastBlockTime;
3
4     // initially contract
5     constructor() {}
6
7     // receive function
8     receive() external payable {}
9
10     // fallback function used to make a bet
11     fallback() external payable {
12         require(msg.value == 1 ether); // must send 1 ether to play
13         // only 1 transaction per block
14         require(block.timestamp != pastBlockTime);
15         pastBlockTime = block.timestamp;
16         if(block.timestamp % 15 == 0) { // winner
17             payable(msg.sender).transfer(address(this).balance);
18         }
19     }
20 }
```

Figure 3: Timestamp dependency vulnerability example.

balance in the vulnerable contract after the transfer operation of call. Value (), then when the attacker calls the transfer function call, value () will trigger the rewritten fallback function of the attacker's contract. The transfer function can be called again to continuously recursively transfer money from the vulnerable contract to the attacker contract. Figure 2 contains an EtherStore contract and an attack contract. The EtherStore contract implements the Ether vault function, which allows users to withdraw one Ether coin per week. Lines 13 and 14 of the EtherStore contract code judge the withdrawal amount and the withdrawal interval, so that a withdrawal can only be performed successfully if the requested withdrawal amount is less than 1 ether and no withdrawals have been made in the last week. Line 15 of the code sends the requested ether to the user via the call.value () function. The attacker launches an attack on the EtherStore contract through the Attack contract and calls the EtherStore contract's withdrawFunds () function to withdraw 1 ether, at which time the requirements of the withdrawal amount and the withdrawal interval are met and the attack contract receives 1 ether from the EtherStore contract and executes the fallback function. In the fallback function, the withdrawFunds () function is called again to reenter the EtherStore contract. When the withdrawFunds () function is called for the second time, lines 16 and 17 of the code have not been executed, and the balance and withdrawal time still meet the contract requirements. Therefore, the attacker can continue to withdraw ether, thus allowing all ether to be withdrawn from the EtherStore contract in a single transaction.

2.3. Timestamp Dependence. Timestamp dependency means that the execution of the smart contract depends on the timestamp of the current block. With the different timestamps, the execution results of the contract also vary. Vulnerabilities arise when blockchain timestamps are used as seeds to generate random numbers or as various time-dependent state change conditional statements to perform certain critical operations [23]. Some variables exist in the block header, including BLOCKHASH, TIMESTAMP, NUMBER, GASLIMIT, and COINBASE, so in principle, they can be influenced by miners. Miners have the right to set block timestamps within a 900 second offset. If the timestamp of the new block is greater than the timestamp of the previous block, and the difference between the timestamps is less than 900 seconds, then the timestamp of the new block is legal. If the cryptocurrency is transmitted based on block variables, a malicious miner could change the timestamps of their blocks to exploit the vulnerability. Figure 3 shows a code example that contains a timestamp dependency vulnerability. The contract implements a lottery function where one transaction per block can be invested with 1 ether, and each player has a 1 in 15 chances of winning the contract

```
contract HashForEther {

    function withdrawWinnings() {
        // Winner if the last 8 hex characters of the address are 0.
        require(uint32(msg.sender) == 0);
        _sendWinnings();
    }

    function _sendWinnings() {
         msg.sender.transfer(this.balance);
     }
}
```

Figure 4: Implicit visibility vulnerability example.

balance according to the contract logic. The vulnerability occurs on line 15, where miners can adjust the block timestamp so that block timestamp takes 15 modulo 0 to win the block reward of ether locked in the contract.

2.4. Implicit Visibility. Functions in Solidity have visibility specifiers, which indicate how the function is called. Visibility determines whether a function can be called externally by the user, by other derived contracts, internally only, or externally only. Functions can be specified as external, public, internal, or private, and incorrect use of visibility specifiers can lead to a number of development vulnerabilities in smart contracts. The default visibility of functions is public, so external users can call functions without specifying any visibility. Vulnerabilities exist when a function is supposed to be private or can only be called within the contract itself, and the developer ignores the function's visibility specifier. The contract in Figure 4 contains an undeclared visibility vulnerability. HashForEther () is an address-guessing bounty game contract that allows users to call the withdrawinnings () function to get their bounty once they generate an Ethereum address with the last 8 hexadecimal characters being 0. However, withdrawinnings () and sendwinnings () do not specify that the function visibility is a public function, so any address can call this function to steal the bounty.

3. Related Work

Smart contract vulnerability detection is one of the fundamental issues of blockchain security. Early methods used static and dynamic analysis methods such as symbolic execution and fuzzing, which depended on expert-defined vulnerability rules [3]. Recent approaches utilize deep learning to build vulnerability detection models that automatically learn vulnerability features. This section will introduce the current research work from two aspects: traditional vulnerability detection methods based on static and dynamic analysis, such as symbol execution and fuzzy testing; and methods based on deep learning.

3.1. Traditional Vulnerability Detection Methods. Slither [24] is the first open source static analysis framework for the Solidity language. Slither converts Solidity smart contracts into an intermediate representation called Slither, which provides fine-grained information about smart contract code and can flexibly support many applications. Smartcheck [25] is an extensible static analysis tool that checks Solidity source code by converting it into an intermediate XML-based representation and then checking it against XPath schemas for comparison. Typical techniques for dynamic analysis include fuzzing and symbolic execution. Oyente [26], a dynamic symbolic execution detection tool, was the first tool for security analysis of smart contracts. ContractFuzzer [27] uses the fuzzing method to detect vulnerabilities in Ethereum smart contracts. It can generate fuzzing input according to the ABI specification of smart contracts and define test predictions for detecting security vulnerabilities. These methods all have the problem of low automation, and most of them need to rely on expert knowledge and human experience to extract fixed rules for vulnerabilities, which is inefficient and laborious. The static analysis method needs to manually set the matching rules according to the analysis efficiency of the matching rules, so it has a high false positive rate. Compared with static code analysis [28–30], dynamic analysis methods execute smart contracts in real blockchain systems, so dynamic analysis methods are more accurate. However, the debugging, analysis, and running of the target program in dynamic analysis requires a large number of personnel to participate, which has some disadvantages, such as slow speed, low efficiency, and the difficulty of carrying out large-scale testing.

3.2. Vulnerability Detection Method Based on Deep Learning. With the continuous development of artificial intelligence technology, there have been many studies applying machine learning, deep learning, and other technologies to the field of vulnerability mining to identify vulnerabilities in smart contracts. Gao [11] learned the structured code embedding of smart contracts based on deep learning methods, collected 52 known vulnerability contracts as a vulnerability database that contains 10 common vulnerabilities, and identified clone defects through the vulnerability database. Xu et al. [15] builds an abstract syntax tree for smart contracts, compares the ASTs of two smart contracts to obtain shared child nodes, and uses structural similarity to detect vulnerabilities. It builds AST of smart contracts from a manually injected vulnerability contract dataset, extracts subnodes shared with marked contracts, and obtains structural similarity through machine learning. However, there are various vulnerability modes in reality, and the contract construction template only by manually injecting vulnerabilities cannot contain all the vulnerability modes in reality. This method of constructing AST templates based on existing vulnerability contracts has poor scalability and can only detect existing vulnerability patterns. Zhuang et al. [31, 32] takes the opcode sequence as the input of the sequence learning model, and uses LSTM to learn the smart contract opcodes to obtain the vulnerability-related features. ContractWard [12] uses the n-gram algorithm to extract binary features from the opcodes of smart contracts to construct feature spaces, and uses machine learning methods to build models for vulnerability detection. Qian et al. [33] constructed a contract graph to represent the syntactic and semantic structure of smart contract functions, and used a graph

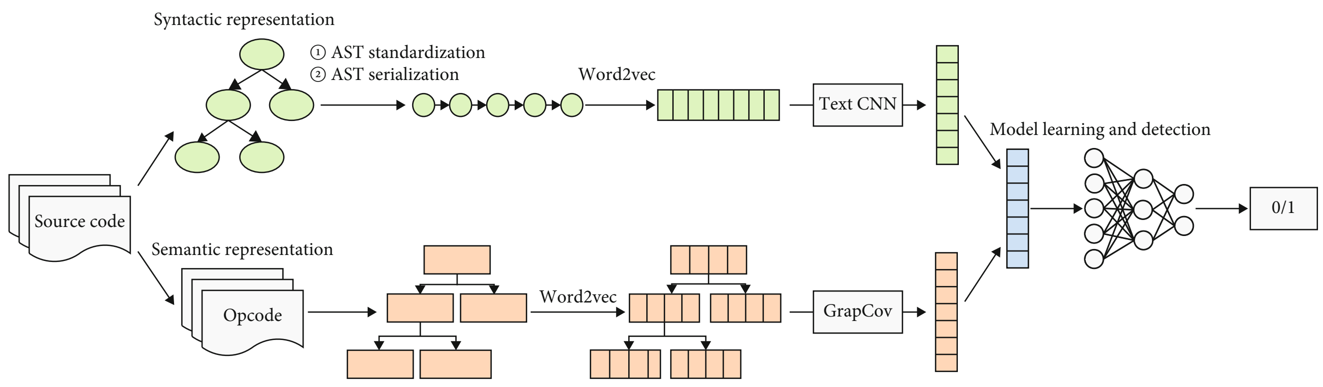
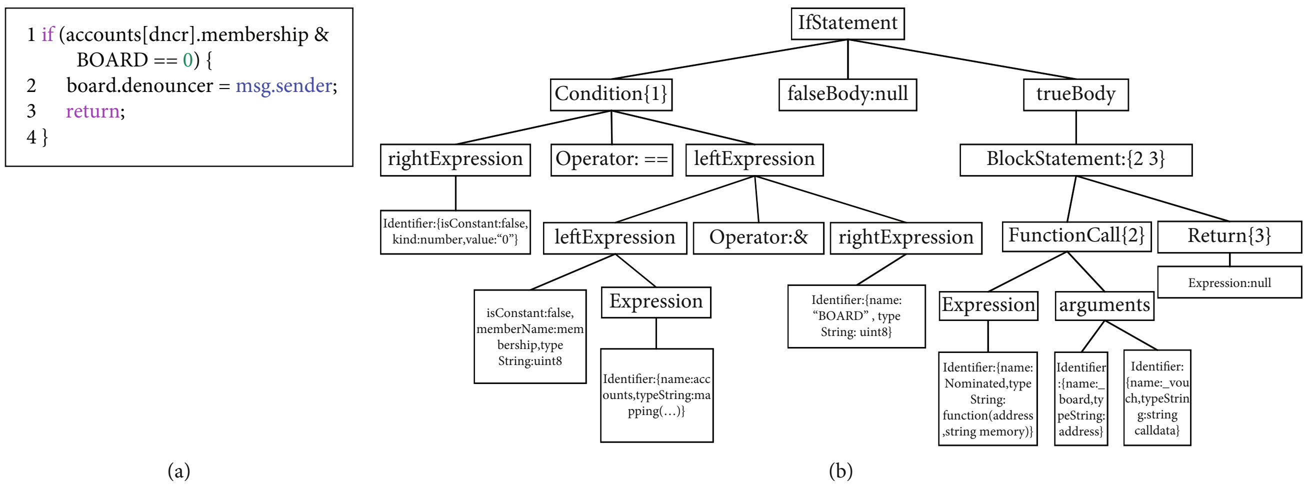


FIGURE 5: Overall framework.



```
if (accounts[dncr].membership &
     BOARD == 0) {
   board.denouncer = msg.sender;
   return;
}
```

IfStatement
Condition{1}
falseBody:null
trueBody
rightExpression
Operator: ==
leftExpression
BlockStatement:{2 3}
Identifier:{isConstant:false, kind:number,value:"0"}
leftExpression
Operator:&
rightExpression
FunctionCall{2}
Return{3}
Expression:null
isConstant:false, memberName:membership,type String:uint8
Expression
Identifier:{name:accounts,typeString:mapping(...)}
Identifier:{name: "BOARD" , type String: uint8}
Expression
arguments
Identifier:{name: Nominated,type String: function(address ,string memory)}
Identifier :{name:_ board,typeString: address}
Identifier: {name:_vouch,typeString:string calldata}
(a)
(b)

FIGURE 6: AST of solidity function (a) a code segment containing if statements. (b) CFG converted from code segment.

convolutional neural network to learn vulnerabilities from the contract graph. Compared with traditional automated vulnerability detection, deep learning-based vulnerability detection methods are more accurate and efficient, and their generalization ability is better. By comparing the current deep learning methods, it can be observed that the key to applying deep learning methods for vulnerability detection lies in the selection of a suitable representation for the source code. Selecting a suitable code representation and then building a feature extractor to extract the features of the source code can further improve the efficiency of vulnerability detection.

4. Method

In order to obtain the multiple features of the contract source code, the model extracts the syntactic features and semantic features from the source code, respectively, and fuses the syntactic and semantic features, using the fusion features to detect vulnerabilities. The overall framework of this paper is shown in Figure 5, which consists of three phases: syntax representation, semantic representation, and model learning and detection. Contract code embedding parses smart contracts into appropriate representation structures for model training. We use AST and CFG to represent smart contracts in two stages to abstract the feature information of source code vulnerabilities. In the syntax representation stage, the AST of the smart contract is obtained from the source code. The AST is normalized and serialized, and then transformed into a vector representation using word embedding. Finally, the syntax features are obtained using Text CNN. In the semantic representation stage, the source code is first compiled into bytecode, and the CFG of the smart contract is obtained from the bytecode. Then use word2vec to vectorize the graph node features and use MPNN to obtain the semantic and structural embeddings of the graph. In the model learning stage, combined with the syntactic vector and semantic vector obtained in the syntactic representation and semantic representation stages, the fully connected layer is used to learn the multifeature vector to train the detector. Finally, the vulnerability detection of smart contract source code is implemented using the trained detector.

4.1. Syntactic Representation. In the syntactic representation stage, we use the AST obtained by compiling the source code

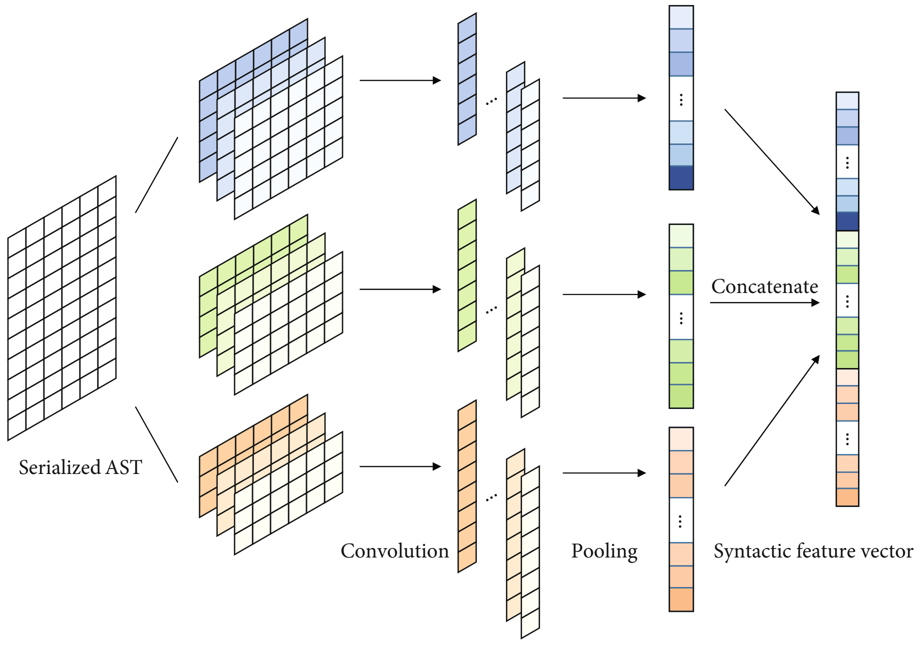


Figure 7: TextCNN model structure.

to represent the syntactic features of the code. AST is a tree representation of the syntax structure of a program's source code, with each node in the tree representing a structure in the source code. This syntax-based code representation can preserve the syntactic structure information of the source code, which represents the syntax and details of each statement, such as the number of functions defined, variable types, etc. Using AST for feature extraction can obtain more syntactic features than source code text.

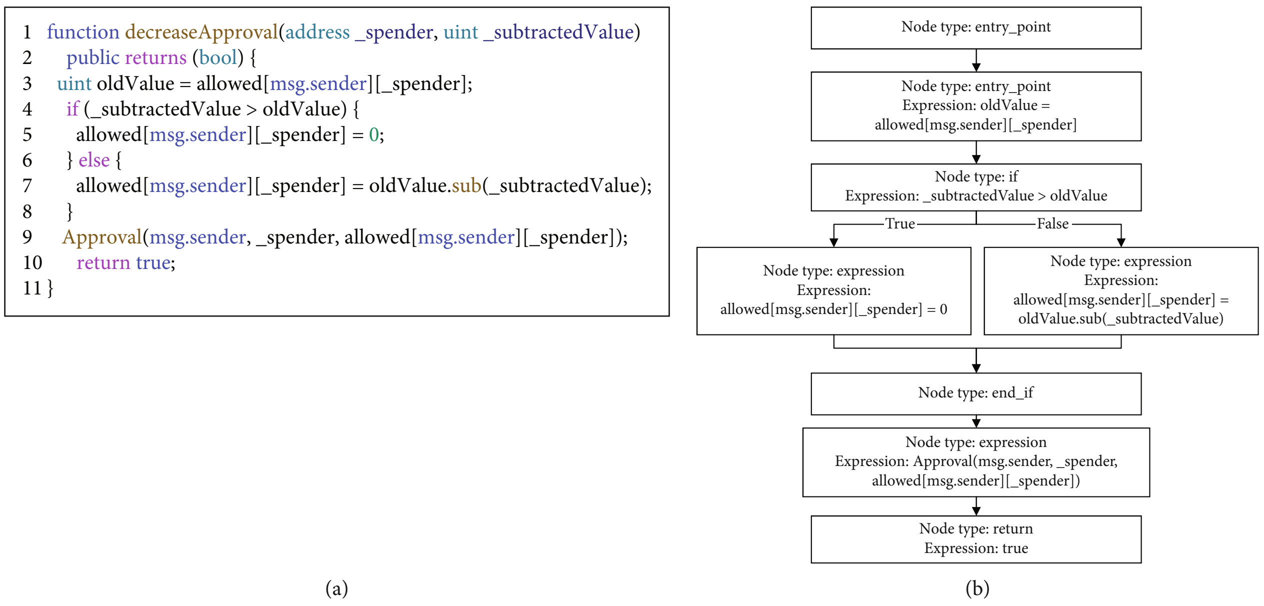
(1) *Get the abstract syntax tree*. First, we obtain the AST of the smart contract. In this paper, we use py-solc-x (a third-party software package for Python) to build the AST of the smart contract. Py-solc-x converts the source code of a smart contract into a tree-structured AST. Figure 6(a) shows a code segment containing if statements. The transformed AST is shown in Figure 6(b), and each node contains information such as type, name, child node, and value. After obtaining the AST, we normalize, serialize, and vectorize the AST

(2) *AST normalization*. The AST contains user-defined contract names, function names, variable names, and variable values. The user-defined names and values vary from contract to contract, which can affect the accuracy of vulnerability detection, leading to more false negatives. To eliminate this variability, we normalized the AST. According to the order in which the user-defined contract name, function name, and variable name appear in the contract, they are, respectively, represented as $(\text{contract } 1, \cdots, \text{contract } n)$, $(\text{function } 1, \cdots, \text{function } n)$, $(\text{var } 1, \cdots, \text{var } n)$. By normalizing, the model can be made to focus more on the contract structure and ignore this variability

(3) *AST serialization*. After normalization, we perform serialization operations on the AST. In order to extract the syntactic structure of the solidity contract, we traverse all nodes of the AST in depth-first order and transform the tree-structured AST into a serially structured code token. The AST constructed from source code is longer than the source code, and in order to be more conducive to model training, we only keep some key information during serialization, while ignoring some secondary information indicating code location and node id

(4) *Vectorization*. Smart contract source code is usually a text representation and cannot be directly used for training deep neural network models. After normalizing and serializing the AST, the AST needs to be represented in vector form as the input for training the model. Each code token is represented as a vector with the same dimension using word2vec [34], thereby representing the serialized AST in vector form. The source code of the contracts varies in length, so the transformed feature vectors have different lengths. We represent a text vector as a fixed-length numeric vector by truncation and padding operations. Pad with zeros when the vector length is less than the fixed length, and truncate when the vector length is greater than the fixed length

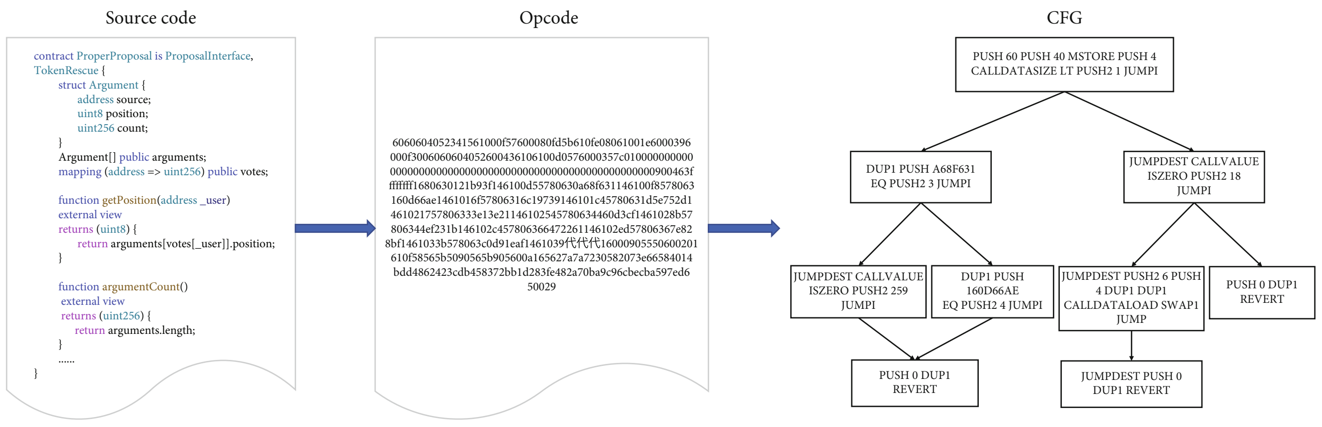
(5) *Syntax Feature Extraction*. Text CNN [35] is a text classification model proposed by Yoon Kim for the deformation of the input layer of CNN, which has many applications in the field of natural language processing [36, 37]. Text CNN makes the model



```
function decreaseApproval(address _spender, uint _subtractedValue)
  public returns (bool) {
 uint oldValue = allowed[msg.sender][_spender];
  if (_subtractedValue > oldValue) {
   allowed[msg.sender][_spender] = 0;
  } else {
   allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
  }
 Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
   return true;
}
```

Node type: entry_point
Node type: entry_point Expression: oldValue = allowed[msg.sender][_spender]
Node type: if Expression: _subtractedValue > oldValue
True
False
Node type: expression Expression: allowed[msg.sender][_spender] = 0
Node type: expression Expression: allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue)
Node type: end_if
Node type: expression Expression: Approval(msg.sender, _spender, allowed[msg.sender][_spender])
Node type: return Expression: true
(a)
(b)

Figure 8: CFG of solidity function (a) code for the decreaseApproval() function. (b) CFG of the function decreaseApproval().



```
contract ProperProposal is ProposalInterface,
TokenRescue {
      struct Argument {
          address source;
          uint8 position;
          uint256 count;
      }
      Argument[] public arguments;
      mapping (address => uint256) public votes;

      function getPosition(address _user)
      external view
      returns (uint8) {
          return arguments[votes[_user]].position;
      }

      function argumentCount()
       external view
       returns (uint256) {
         return arguments.length;
      }
      ......
}
```

```
6060604052341561000f57600080fd5b610fe08061001e6000396
000f300606060405260043610610 0d0576000357c010000000000
00000000000000000000000000000000000000000000900463f
ffffff1680630121b93f146100d55780630a68f631146100f8578063
160d66ae1461016f57806316c19739146101c45780631d5e752d1
461021757806333e13e2114610254578063446 0d3cf1461028b57
806344ef231b146102c45780636647226114610 2ed57806367e82
8bf1461033b578063c0d91eaf1461039代代代16000905550600201
610f58565b5090565b905600a165627a7a7230582073e66584014
bdd4862423cdb458372bb1d283fe482a70ba9c96cbecba597ed6
50029
```

Source code
Opcode
CFG
PUSH 60 PUSH 40 MSTORE PUSH 4 CALLDATASIZE LT PUSH2 1 JUMPI
DUP1 PUSH A68F631 EQ PUSH2 3 JUMPI
JUMPDEST CALLVALUE ISZERO PUSH2 18 JUMPI
JUMPDEST CALLVALUE ISZERO PUSH2 259 JUMPI
DUP1 PUSH 160D66AE EQ PUSH2 4 JUMPI
JUMPDEST PUSH2 6 PUSH 4 DUP1 DUP1 CALLDATALOAD SWAP1 JUMP
PUSH 0 DUP1 REVERT
PUSH 0 DUP1 REVERT
JUMPDEST PUSH 0 DUP1 REVERT

Figure 9: Control flow graph transformation process.

work well by introducing the trained word vectors, and can solve the problem of gradient disappearance of long texts. We use Text CNN to extract syntax features, and the model structure is shown in Figure 7. We define multiple 1D convolution kernels and perform convolution operations on the input separately. Convolution kernels of different widths can capture local features between different numbers of adjacent tokens. Adaptive average pooling is performed on all output channels, then all scalar convergence outputs are concatenated into vectors, and finally we obtain the syntactic representation of the smart contract

4.2. Semantic Representation. In the semantic representation stage, the CFG of each contract is obtained to extract the semantic features of the contract code. CFG is an abstract representation of a process or program, an abstract data structure used in a compiler that represents all the paths that a program will traverse during its execution. Each node in the CFG represents a basic block, that is, a piece of code without any jumps or jump targets, which is process-oriented and can reflect many information and execution flows of all basic blocks in a process. Figure 8(a) shows a contract function decreaseApproval() containing an if statement, and Figure 8(b) shows the CFG that the function is transformed into, displaying the execution flow of the function's statement. Through CFG, it is possible to traverse all the execution paths of a function, and discover the control dependencies and data dependencies that exist in the program. The data dependencies and control dependencies embodied by CFG are the missing information in the AST, which can capture more comprehensive semantic information, and thus the code semantic representation with CFG can complement the syntactic features extracted by AST. The semantic representation of code with CFG can supplement the syntactic features extracted by AST.

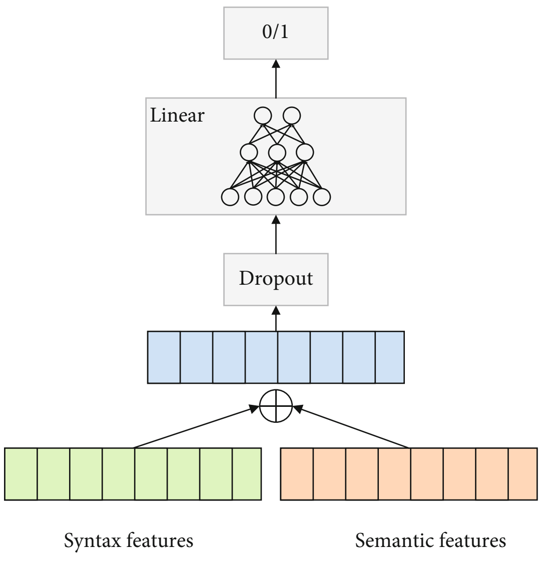


Figure 10: Feature fusion.

Table 1: Number of dataset samples.

Vulnerability type	No vulnerability	Vulnerability	Proportion
Implicit visibility	11677	11200	1.04 : 1
Integer overflow	14114	8763	1.61 : 1
Integer underflow	19236	3614	5.32 : 1
TimeDependency	22552	325	69.3 : 1
Reentrancy	22814	63	362 : 1

(1) *Get the control flow graph*. In the conversion of smart contracts to control flow graphs, the smart contracts are first compiled into opcodes, and then the opcodes are converted into control flow graphs by evm_cfg_builder. The converted CFG is shown in Figure 9, where each node is an independent block of code and the statements in the node are represented by opcodes

(2) *Block embedding*. In the processing of CFG, semantic information and structural information are obtained through two steps: block embedding and graph embedding. Each block in the CFG contains the statement information, and the block embedding through word2vec can preserve the opcode information of the source code. The statements in each block are represented as a fixed-length vector after block embedding, and the semantic vector is used as the feature of the graph node

(3) *Graph embedding*. After the node features of the CFG are obtained through block embedding, the CFG is represented by a graph neural network to obtain a graph embedding vector representing the semantics and structure of the entire CFG graph. Graph Convolutional Neural Networks [38] apply convolutional operations to graph data, aggregating node own features and domain node features to generate new node representations. In this paper, we use a Graph Convolutional Neural Network to update the CFG node representation and use the mean-pool readout layer to obtain the graph-embedding vector. Through block embedding and graph embedding, the final graph embedding representation preserves the overall structural information of the CFG, as well as the semantic information of each node's opcode

4.3. Model Learning and Vulnerability Detection. After obtaining the syntactic embedding and semantic embedding of the source code through the syntactic characterization and semantic characterization stages, the syntactic and semantic feature vectors are fused for model training. In this paper, we choose to perform vulnerability detection at the contract level. Since the AST obtained in the syntactic representation stage is at the file level, and the CFG obtained in the semantic representation is at the contract level, it is necessary to split the entire AST into multiple small contract ASTs before splicing the feature vectors. The fused feature vectors combined with source code syntax and semantic information can retain more feature information in the model learning stage. As shown in Figure 10, we fuse syntactic and semantic feature vectors as input to train a feedforward neural network for vulnerability detection. We take the obtained fused vector as the input to the model to automatically learn the potential code features from the fused feature vector. Finally, the learned vulnerability detection model is used in the vulnerability detection stage.

After training to obtain a vulnerability detection model, vulnerability detection can be performed on smart contracts. The process of the detection stage is the same as that of the learning stage in terms of data preprocessing and data representation. After the target program is vectorized by syntax representation and semantic representation, it is fed into the vulnerability inspection model obtained in the learning stage to obtain the prediction result. The method in this paper supports the detection of solidity source code and can directly detect the contract source code. When detecting smart contracts, the test contract is inputted into the embedding module of syntactic and semantic representation, and any given new smart contract is transformed into a vector representation by obtaining the AST and CFG of the contract and utilizing the learned embedding matrix. The smart contract is then detected using the vulnerability detection model obtained from the training to determine whether there is a vulnerability ("1") or no vulnerability ("0").

5. Experiments

In this section, we describe the vulnerability dataset used in this paper, analyze the ratio of vulnerable to nonvulnerable codes in the dataset, and conduct extensive experiments on the test set to evaluate the effectiveness of the model proposed in this paper. First, the performance of the detection model is measured using accuracy, precision, recall, and F1 metrics to illustrate the overall performance of our model. Secondly, the effectiveness of TextCNN for extracting

Table 2: Dataset experimental results.

	Methods	Accuracy	Precision	Recall	F1
ImplicitVisibility	Conkas	—	—	—	—
	Opcode	54.16	61.86	21.73	32.16
	Our model	**89.00**	**87.58**	**92.04**	**89.75**
IntegerOverflow	Conkas	56.63	24.64	87.51	38.45
	Opcode	83.28	81.71	85.75	83.68
	Our model	**95.58**	**94.08**	**97.79**	**95.90**
IntegerUnderflow	Conkas	87.01	63.09	72.76	67.58
	Opcode	72.70	71.61	75.20	75.48
	Our model	**96.42**	**92.86**	**88.51**	**90.63**
TimeDependency	Conkas	92.97	57.64	67.49	62.18
	Opcode	94.04	97.01	90.89	93.85
	Our model	**98.25**	**97.52**	**94.85**	**94.81**
Reentrancy	Conkas	83.02	49.80	70.22	58.27
	Opcode	87.51	81.48	**97.08**	**88.60**
	Our model	**98.64**	**90.91**	78.38	75.36

syntactic features is verified. Finally, the single-feature code representation and multifeature code representation are compared to verify the effectiveness of the multifeature vulnerability detection proposed in this paper.

5.1. Datasets and Data Preprocessing. This paper uses the Eth2Vec [39] dataset for model training. The Eth2Vec dataset contains 5,000 smart contract files of the real Ethereum environment, with a total of 22,879 contract functions. The labeling results of the five vulnerability types in the Eth2Vec dataset are shown in Table 1, and it can be observed that the number of positive and negative samples in the dataset is unbalanced. The number of positive and negative samples for Implicit Visibility and Integer Overflow vulnerabilities is relatively balanced, while the proportion of positive and negative samples for the other three types of vulnerabilities differs significantly. The datasets with reentrancy vulnerabilities and timestamp dependencies are the most unbalanced, with less than 400 positive examples. The unbalanced dataset will affect the training results of the model, which is not conducive to the model learning the features of classes with fewer samples. Therefore, we preprocess the dataset to alleviate the data imbalance problem. In this paper, we use a weighted random sampling method to process the dataset to balance the number of samples.

5.2. Results and Analysis. In this section, we illustrate the experimental results and performance comparisons of our model on smart contract vulnerability detection. For a dataset with few positive samples, the model may detect all samples as negative samples to obtain a higher accuracy, so we should also pay attention to the recall metric while focusing on the accuracy. The recall metric represents the ratio of predicted correct positive samples to true positive samples, reflecting the ability of the model to effectively detect vulnerabilities. We evaluate the effectiveness of the model with accuracy, precision, recall, and F1-score.

5.2.1. Overall Performance. In order to verify the performance of the model in this paper, we compared it with the current detection methods, and the experimental results are shown in Table 2. First, we compare it with traditional vulnerability detection methods. Conkas [40] is a smart contract static analysis tool that detects vulnerabilities using symbolic execution. Conkas can detect vulnerability types such as arithmetic, timestamp dependency, and reentrancy vulnerability. Second, we compare it with deep learning-based detection methods. Opcode [31] detects smart contract security vulnerabilities at the opcode level, so we first compile the contract into opcodes and then detect the opcodes. Compared with Conkas and Opcode, our model achieves better performance on all five detection tasks. In the reentrancy vulnerability detection, there are only 63 contracts with vulnerabilities, with a large difference in the number of positive and negative samples. Although balanced by weighted random sampling, there are still too few vulnerability features that can be learned, so the recall for reentrancy vulnerability detection is low compared to other vulnerability types. The recall rates of Integer Overflow and Integer Underflow vulnerabilities reached 97.79% and 88.51%, with performance improvements of 10.28% and 13.31%, respectively. The experimental results illustrate the effectiveness of our model.

5.2.2. Validity of Syntactic Representation. To verify the effectiveness of the syntactic representation module for AST syntactic feature extraction, we compared it with LSTM-AST and GRU-AST. We replace the syntactic feature extraction part of the model with LSTM and RNN, while the other parts of the model remain unchanged. LSTM-AST indicates that the feature extraction part of the syntactic representation is replaced by LSTM to extract the syntactic features of AST. GRU-AST indicates that the syntactic features of AST are extracted by GRU [41]. In the syntactic representation of the source code, we chose Text CNN for syntactic

Table 3: Syntax representation module experimental results.

	Methods	Accuracy	Precision	Recall	F1
ImplicitVisibility	LSTM-AST	85.50	84.82	88.06	86.41
	GRU-AST	86.24	85.32	86.46	86.95
	Our model	**89.00**	**87.58**	**92.04**	**89.75**
IntegerOverflow	LSTM-AST	94.17	93.52	95.58	94.54
	GRU-AST	93.92	92.28	95.05	94.16
	Our model	**95.58**	**94.08**	**97.79**	**95.90**
IntegerUnderflow	LSTM-AST	96.17	92.76	87.23	89.91
	GRU-AST	95.83	89.70	**88.94**	89.32
	Our model	**96.42**	**92.86**	88.51	**90.63**
TimeDependency	LSTM-AST	94.29	85.48	88.79	87.10
	GRU-AST	95.92	89.61	89.22	89.42
	Our model	**98.25**	**97.52**	**94.85**	**94.81**
Reentrancy	LSTM-AST	97.83	68.97	54.05	60.61
	GRU-AST	97.92	63.64	70.27	69.14
	Our model	**98.64**	**90.91**	**78.38**	**75.36**

Table 4: Comparison of experimental results of each module.

		Accuracy	Precision	Recall	F1
ImplicitVisibility	CFG	86.08	85.96	87.74	86.84
	AST	88.00	**92.76**	83.60	87.94
	AST + CFG	**89.00**	87.58	**92.04**	**89.75**
IntegerOverflow	CFG	92.92	91.03	96.06	93.48
	AST	94.08	93.37	95.58	94.47
	AST + CFG	**95.58**	**94.08**	**97.79**	**95.90**
IntegerUnderflow	CFG	95.25	91.59	83.40	87.31
	AST	94.66	90.91	80.85	85.59
	AST + CFG	**96.42**	**92.86**	**88.51**	**90.63**
TimeDependency	CFG	95.42	89.33	86.64	87.96
	AST	98.25	96.48	92.24	**95.42**
	AST + CFG	**98.25**	**97.52**	**94.85**	94.81
Reentrancy	CFG	97.33	58.06	48.65	52.94
	AST	98.58	76.32	73.68	**77.33**
	AST + CFG	**98.64**	**90.91**	**78.38**	75.36

feature extraction, considering that the AST sequence transformed into the source code is long and the word order information is not obvious. The experimental results are shown in Table 3. The syntactic representation module achieves optimality in all four metrics for the four vulnerability types of detection of ImplicitVisibility, IntegerOverflow, IntegerUnderflow, TimeDependency, and Reentrancy. The syntax characterization module improves precision and recall by 2%-7% on ImplicitVisibility, IntegerOverflow, and IntegerUnderflow vulnerability detection compared to the replaced model. The precision of TimeDependency and Reentrancy vulnerability detection has improved by 8% and 22%, and the recall rate has improved by 5% and 8%. In IntegerUnderflow vulnerability detection, although the syntactic representation module has a slightly lower recall than GRU-AST, but the performance of the other three metrics is improved, and it obtains a more balanced performance. The experimental results show that the syntactic representation module can effectively extract the syntactic features of the source code. Compared with the sequence model, its performance is more balanced and stable in syntactic feature extraction for AST. We observe that the sequence model focuses more on the sequence order information, but the AST sequence does not have significant temporal order information. TextCNN defines different filters to extract source code features from different perspectives, insensitive to word order information, and more suitable for AST sequences. Therefore, the syntactic features of the source code can be obtained more efficiently using TextCNN.

5.2.3. Effectiveness of Fusion Code Representation. We compare the single code representation with the multifeature code representation to verify the validity of each module and the validity of the fused code representation. To verify the validity of the syntactic representation module, we only use the AST representation of the source code for vulnerability detection. To verify the validity of the semantic representation module, only the semantic representation module is used for feature extraction of CFG. The experimental results are shown in Table 4. By comparing the four metrics, the vulnerability detection performance using AST is slightly better than CFG. This is because a part of the feature information is lost in the semantic representation module when representing the semantic information of each block as a fixed-length feature vector. In addition, the multifeature code representation improves the most in the recall metric, and the model has higher performance and accuracy in correctly identifying vulnerable codes. The highest accuracy and recall are achieved by fusion learning to use AST and CFG, which validates that our proposed syntactic representation and semantic representation modules can capture more features of source code for vulnerability detection.

6. Conclusions

In this paper, we propose a new approach for smart contract vulnerability detection, which characterizes the code by fusion learning of syntactic and semantic information. The syntactic and semantic features in the source code are automatically learned through the syntactic representation and semantic representation modules, and the syntactic-semantic fusion vectors are used to detect smart contract vulnerabilities. We conducted experiments on the smart contract dataset in Ethereum, and the results show that our method can accurately identify multiple types of vulnerabilities, with significant improvements in precision and recall compared to existing methods. Our model detects against contract source code without defining complex vulnerability patterns, enabling effective and automated vulnerability detection. In the following work, we will explore more effective vulnerability detection models that can localize the location of the vulnerable code while accurately detecting the vulnerability.

Data Availability

Previously reported smart contract source code data were used to support this study and are available at https://github.com/fseclab-osaka/eth2vec. These prior studies and datasets are cited at relevant places within the text as references Eth2Vec.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This research was funded by the National Natural Science Foundation of China (41871316), the Scientific and technological project in Henan Province (212102210414 and 22B520003), and the Foundation of University Young Key Teacher of Henan Province (Grant No. 2019GGJS040 and 2020GGJS027).

References

[1] N. Szabo, “Smart contracts: building blocks for digital markets,” 1996, https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html.

[2] X. B. Wang, X. Y. Yang, X. F. Shu, and L. Zhao, “Formal verification of smart contracts based on MSVL,” *Journal of Software*, vol. 6, no. 32, pp. 1849–1866, 2021.

[3] Z. H. A. N. G. Ying-li, M. A. Jia-li, L. I. U. Zi-ang, L. I. U. Xin, and Z. H. O. U. Rui, “Overview of vulnerability detection methods for Ethereum solidity smart contracts,” vol. 49, Tech. Rep. 3, Computer Science, 2022.

[4] H. Feng, X. Fu, H. Sun, H. Wang, and Y. Zhang, “Efficient vulnerability detection based on abstract syntax tree and deep learning,” in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 722–727, Toronto, ON, Canada, 2020.

[5] L. Lv, Z. Wu, J. Zhang, L. Zhang, Z. Tan, and Z. Tian, “A VMD and LSTM based hybrid model of load forecasting for power grid security,” *IEEE Transactions on Industrial Informatics*, vol. 18, no. 9, pp. 6474–6482, 2022.

[6] S. D. Okegbile, J. Cai, C. Yi, and D. Niyato, “Human digital twin for personalized healthcare: vision, architecture and future directions,” *IEEE Network*, pp. 1–7, 2022.

[7] D. Cao, J. Huang, X. Zhang, and X. Liu, “FTCLNet: convolutional LSTM with fourier transform for vulnerability detection,” in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 539–546, Guangzhou, China, 2020.

[8] C. Yi, J. Cai, and Z. Su, “A multi-user mobile computation offloading and transmission scheduling mechanism for delay-sensitive applications,” *IEEE Transactions on Mobile Computing*, vol. 19, no. 1, pp. 29–43, 2020.

[9] K. B. Kim and J. Lee, “Automated generation of test cases for smart contract security analyzers,” *IEEE Access*, vol. 8, pp. 209377–209392, 2020.

[10] J. Qiu, Z. Tian, C. Du, Q. Zuo, S. Su, and B. Fang, “A survey on access control in the age of internet of things,” *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4682–4696, 2020.

[11] Z. Gao, “When deep learning meets smart contracts,” in *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pp. 1400–1402, Virtual Event, Australia, 2020.

[12] W. Wang, J. Song, G. Xu, Y. Li, H. Wang, and C. Su, “ContractWard: automated vulnerability detection models for Ethereum

smart contracts," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1133–1144, 2021.

[13] L. Zhang, Z. Huang, W. Liu, Z. Guo, and Z. Zhang, "Weather radar echo prediction method based on convolution neural network and long short-term memory networks for sustainable e-agriculture," *Journal of Cleaner Production*, vol. 298, article 126776, 2021.

[14] L. Lv, Z. Wu, L. Zhang, B. B. Gupta, and Z. Tian, "An edge-AI based forecasting approach for improving smart microgrid efficiency," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 11, pp. 7946–7954, 2022.

[15] Y. Xu, G. Hu, L. You, C. Cao, and A. Derhab, "A novel machine learning-based analysis model for smart contract vulnerability," *Security and Communication Networks*, vol. 2021, Article ID 5798033, 12 pages, 2021.

[16] L. Zhang, Y. Huo, Q. Ge, Y. Ma, Q. Liu, and W. Ouyang, "A privacy protection scheme for IoT big data based on time and frequency limitation," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 5545648, 10 pages, 2021.

[17] C. Qiao, K. Brown, F. Zhang, and Z. Tian, "Federated adaptive asynchronous clustering algorithm for wireless mesh networks," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[18] Y. Sun, Z. Tian, M. Li, S. Su, X. Du, and M. Guizani, "Honeypot identification in softwarized industrial cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5542–5551, 2021.

[19] J. Feist, G. Greico, and A. Groce, "Slither: a static analysis framework for smart contracts," in *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, pp. 8–15, Montreal, QC, Canada, 2019.

[20] C. Yi, J. Cai, T. Zhang, K. Zhu, B. Chen, and Q. Wu, "Workload re-allocation for edge computing with server collaboration: a cooperative queueing game approach," *IEEE Transactions on Mobile Computing*, p. 1, 2021.

[21] Y. Sun, Z. Tian, M. Li, C. Zhu, and N. Guizani, "Automated attack and defense framework toward 5G security," *IEEE Network*, vol. 34, no. 5, pp. 247–253, 2020.

[22] L. Zhang, C. Xu, Y. Gao, Y. Han, X. du, and Z. Tian, "Improved Dota2 lineup recommendation model based on a bidirectional LSTM," *Tsinghua Science and Technology*, vol. 25, no. 6, pp. 712–720, 2020.

[23] L. Lv, J. Chen, Z. Zhang, B. Wang, and L. Zhang, "A numerical solution of a class of periodic coupled matrix equations," *Journal of the Franklin Institute*, vol. 358, no. 3, pp. 2039–2059, 2021.

[24] S. Tikhomirov, E. Voskresenskaya, I. Ivanitskiy, R. Takhaviev, E. Marchenko, and Y. Alexandrov, "Smart check: static analysis of Ethereum smart contracts," in *Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain*, pp. 9–16, Gothenburg, Sweden: Association for Computing Machinery, 2018.

[25] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 254–269, Vienna, Austria: Association for Computing Machinery, 2016.

[26] B. Jiang, Y. Liu, and W. K. Chan, "Contract fuzzer: fuzzing smart contracts for vulnerability detection," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering: Association for Computing Machinery*, pp. 259–269, Montpellier, France, 2018.

[27] W. J.-W. Tann, X. J. Han, S. Sengupta, and Y. J. A. Ong, "Towards safer smart contracts: a sequence learning approach to detecting vulnerabilities," 2018, https://arxiv.org/abs/1811.06632.

[28] C. Qiao, J. Qiu, Z. Tan, G. Min, A. Y. Zomaya, and Z. Tian, "Evaluation mechanism for decentralized collaborative pattern learning in heterogeneous vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2022.

[29] L. Zhang, S. Tang, and L. Lv, "An finite iterative algorithm for sloving periodic Sylvester bimatrix equations," *Journal of the Franklin Institute*, vol. 357, no. 15, pp. 10757–10772, 2020.

[30] C. Yi, J. Cai, K. Zhu, and R. Wang, "A queueing game based management framework for fog computing with strategic computing speed control," *IEEE Transactions on Mobile Computing*, vol. 21, no. 5, pp. 1537–1551, 2022.

[31] Y. Zhuang, Z. Liu, P. Qian, Q. Liu, X. Wang, and Q. He, "Smart contract vulnerability detection using graph neural networks," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence; Yokohama*, Yokohama, Japan, 2021.

[32] L. Lv, S. Tang, and L. Zhang, "Parametric solutions to generalized periodic Sylvester bimatrix equations," *Journal of the Franklin Institute*, vol. 357, no. 6, pp. 3601–3621, 2020.

[33] P. Qian, Z. G. Liu, Q. M. He, B. T. Huang, D. Z. Tian, and X. Wang, "Smart contract vulnerability detection technique: a survey," *Journal of Software*, vol. 33, no. 8, pp. 3059–3085, 2022.

[34] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2*, pp. 3111–3119, Lake Tahoe, Nevada: Curran Associates Inc, 2013.

[35] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751, Doha, Qatar, Association for Computational Linguistics, 2014.

[36] C. Yi, S. Huang, and J. Cai, "Joint resource allocation for device-to-device communication assisted fog computing," *IEEE Transactions on Mobile Computing*, vol. 20, no. 3, pp. 1076–1091, 2021.

[37] L. Lv, J. Chen, L. Zhang, and F. Zhang, "Gradient-based neural networks for solving periodic Sylvester matrix equations," *Journal of the Franklin Institute*, vol. 359, no. 18, pp. 10849–10866, 2022.

[38] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017, https://arxiv.org/abs/1609.02907.

[39] N. Ashizawa, N. Yanai, J. P. Cruz, and S. Okamura, "Eth2Vec: learning contract-wide code representations for vulnerability detection on Ethereum smart contracts," in *Proceedings of the 3rd ACM International Symposium on Blockchain and Secure Critical Infrastructure*, pp. 47–59, Virtual Event, Hong Kong, 2021.

[40] N. Veloso, Ed., "Conkas: a modular and static analysis tool for Ethereum bytecode," 2021, https://fenix.tecnico.ulisboa.pt/downloadFile/1689244997262417/94080-Nuno-Veloso_resumo.pdf.

[41] K. Cho, B. van Merrienboer, C. Gulcehre et al., "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar. Association for Computational Linguistics, 2014.

Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 2515962, 8 pages
https://doi.org/10.1155/2022/2515962

Research Article

Modeling and Analysis of Group User Portrait through WeChat Mini Program

Guangmin Li,[1,2] **Wenjing Chen**,[3] **Xiaowei Yan**,[4] **and Li Wang**[1]

[1]*College of Computer Information Engineering, Hubei Normal University, 435002, China*
[2]*College of Arts and Science, Hubei Normal University, 435002, China*
[3]*High-Tech Development Promotion Center Huangshi, 435002, China*
[4]*School of Computer Science, China University of Geosciences, 430074, China*

Correspondence should be addressed to Wenjing Chen; 1269639837@qq.com and Xiaowei Yan; kkuma7@outlook.com

Received 26 June 2022; Accepted 9 August 2022; Published 2 September 2022

Academic Editor: Lingling Lv

Meeting users' preferences and increasing business revenue is an ongoing challenge in the mobile service application. In this paper, we address these challenges by mining mobile user behavior patterns and propose an approach to construct a group user portrait by analyzing access data collected from the users of the WeChat Mini Program. We extract the attributes of mobile users considering their geographic information, online duration, and age group. Using Z-score standardized processing and K-means clustering algorithm, we then model the user portraits through three dimensions including daily average duration, interaction intensity, and access frequency. Our analysis has two important features. Firstly, the significant log data used in our experiments was collected from the production environment ensuring that the results reflect the real attributes of WeChat Mini Program users' behavior. Secondly, we provide data-driven decision-making to help marketers enhance the quality of the product and improve user experience. The experimental results indicate that by distilling and analyzing the key factors from the log data, the characters of typical users can be properly profiled to help product owners better optimize the exact set of the features which need to sustain and further grow.

1. Introduction

In recent years, mobile Internet applications have been extensively developed and the activity profile of mobile users has been significantly changed. The corresponding available event log data includes a wealth of information regarding the user behavior which can be mined to obtain useful insights for commercial incorporations. Such insights can be also used to dramatically improve user's experience and unearth hidden revenue opportunities, either through enhanced performance, customized user interface, or targeted advertisements.

WeChat Mini Program represented by Tencent is dominating the mobile ecosystem in China. According to the WeChat Mini Program Official Report, the daily number of active users (DAU) exceeded 400 million, and the average monthly use time of the Mini Program was 64 minutes as of March 2020. The permeability of the active users of the Mini Program accounted for 78.9%. Competition in this market has also recently increased. Therefore, one of the main challenges is finding new ways to effectively improve the functions of the product and maintain a high customer retention rate.

User portrait, namely, user profiling, refers to acquiring, extracting, and representing the features of the user in the form of a rich semantic-based structure [1, 2]. User profiling includes basic demographic information (such as name, gender, and nationality) as well as dynamic behavior information (such as interests and preferences) [3]. User portrait is widely used in several research fields. For instance, Ontika et al. [4, 5] presented a machine learning method to realize the identity of lyrics authors through user portraits. Also, Xu et al. [6] collected Douban movie data and the content of users' comments to design a social media resource aggregation model. They then used this model to establish a mapping relationship between the user portrait and the resource portrait, providing a reference for resource aggregation. Zhang et al. [7] also extracted the characteristics of a group of the paid user group and provided a three-dimensional user attribute based on their viewing data to retain the core users.

In the traditional retail industry, Gu et al. [8] proposed a psychological modeling method to profile the big five personality traits of the users with their emotion-bearing tweets to accordingly customize their personalized services.

Although there are many research works on the user portrait, there are still research gaps especially on the behavioral data generated from WeChat Mini Program. To the best of our knowledge, there are no systematic studies on the user portrait to profile the target users and provide decision support for the companies.

In this paper, we derive insights on identifying potential users and the importance of understanding different users' motivations and concerns. This paper utilizes log datasets including mobile users' behavior and mine the deep-level representative implicit information and outlines the specific groups of the user portrait. The datasets are collected from a medium-sized application called EnglishMyName and collected from September 5, 2018, to November 24, 2019. The multidimensional log data consists of geographical distribution, online duration, and user term query.

There are three main contributions in this paper: (1) Inferring the users' behavior characteristics and static attributes using mobile clickstream data; (2) building and deploying a WeChat Mini Program application to collect simultaneously behavior data online in compliance with the user privacy; and (3) providing practical suggestions for product operators and service providers using clustering technology and user portrait features.

The rest part of this paper is organized as follows: The related work is reviewed in Section 2. Section 3 introduces experimental preparation and data processing. Section 4 presents cluster analysis, compares four internal validation measures, and also profiles the groups of user portraits. Finally, the conclusions are drawn, and future research direction is discussed in Section 5.

2. Related Work

Alan Cooper, the Father of Interaction Design, introduced the concept of user portrait which is considered to be a fictional, specific, and concrete representation of the target user [9]. In the era of big data, massive data can reflect on user intentions, implying the user behavior patterns and interest preferences. Therefore, user portrait has been extensively investigated in recent years. Alan et al. [10] analyzed Twitter users from three perspectives, geographic location, gender, and belief, and found that the Internet users truly reflect the true population distribution in every area of the USA. Ruas et al. [11] identified different user behaviors through clustering methods based on the degree of interaction among Facebook users. They classified the users into three types: audience, participant, and content producer. Yu et al. [12, 13] also proposed variant regression algorithms to model mobile user gender and personality traits from their mobile phone sensory data. Based on the hotel review data, Shan et al. [14] constructed a user portrait from three dimensions, user information attribute, hotel information attribute, and user evaluation information attribute, and then provided the basis for merchants to understand customers' needs through precision marketing. Zeng et al. [15] conducted star fans' user portraits from social media topic data collected from Sina Weibo and dug out the targeted fans groups. This can help enterprises better tailor their marketing efforts. Liu et al. [16] concluded that the individual user portrait research is focused on specific users in a certain scene and labeled them with multifaceted features. It is suitable for distinguishing different users but not for exploring the user's behavior regulations in groups.

Furthermore, Akbari et al. [17] proposed that the group user portrait research can highlight users' behavior patterns and categorize them into different groups by clustering and association rules. The existing studies have modeled the user portrait based on the datasets mainly collected from social media platforms such as Twitter, Facebook, Weibo, and e-commerce platforms. However, few researchers paid attention to the user portrait modeling techniques on the WeChat Mini Program application. To bridge this gap, here, we explore geography distribution, online duration, and user query preference to mine the behavioral patterns of the mobile users.

Here, we use the K-means algorithm for clustering mainly due to its simplicity and efficiency [18–22]. Clustering here is used to categorize mobile visitors based on their behavioral data and formulate their corresponding relevant marketing strategies. Additionally, a considerable amount of the previous works on the behavioral attributes for user profiling inspired us to have a comprehensive group user profiling type [21, 23]. Considering the typical business scenarios, we further extract three behavioral features (i.e., daily average duration, interaction intensity, and access frequency) for user portrait inference to make clustering analysis results and decision more applicable.

3. Data Preprocessing and Data Analysis

In this section, we collect log items from the WeChat Mini Program app, EnglishMyName under a strict privacy policy. This app aims to serve Chinese users who pick a transliterated English name for their given name. The entire dataset contains 515, 684 items that occurred from September 5, 2018, to November 24, 2019, and are saved as CSV files. The files consist of user identification number, request time, operation type, and request content. The log list allows us to learn which page content has been accessed, at what time, where, and by whom.

The original request content is in an arbitrary format, but it must be machine-readable. To achieve it, we use data preprocessing which is critical for performing user behavior analysis before performing the data mining task. It includes data cleaning, data transformation, and data reduction.

The process of data cleaning is to remove noisy or irrelevant data. Specifically speaking, if a user just logs in to this app with no further action, the landing records are

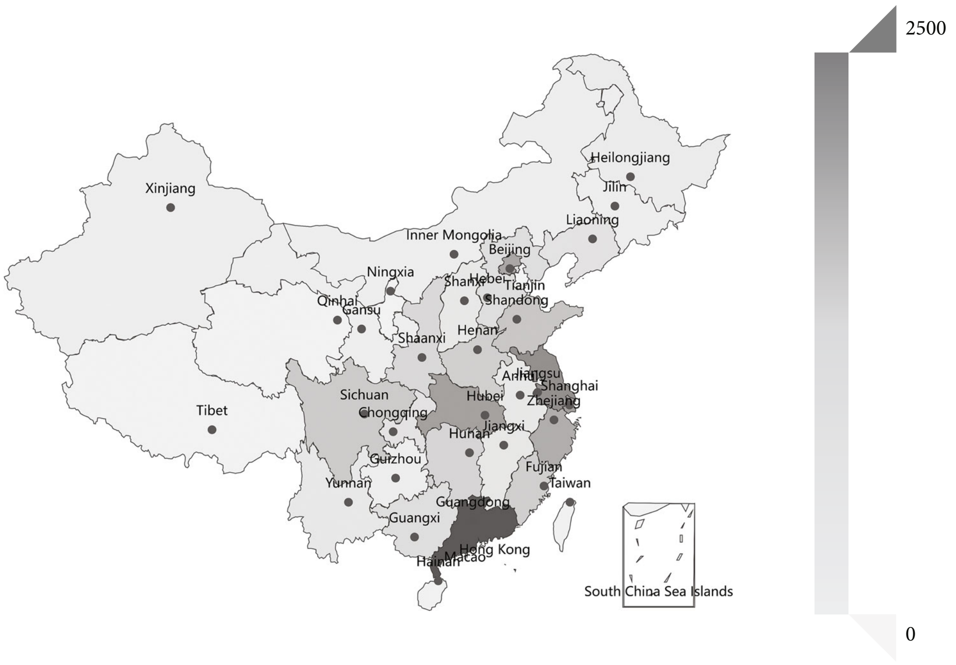


Figure 1: Distribution of the users' locations.

eliminated. During the development and testing phase, huge amounts of debug records are generated, and we eliminate them from the raw logs since they are irrelevant. Furthermore, automated programs like web crawlers are removed from the log files. On the other hand, the data format which is stored in chronological order must be changed into the tabular form that is at a relatively fine granularity. In the last phrase, the structured data is processed with the technique called Z-score normalization. Finally, we selected 19,383 records on online user behavior and used these resources to model and analyze the user portrait.

For one mobile application, each user action in one session shows differentiated performance. Inspired by the user information to profile [24–28], this paper illustrates mobile user behavior from multidimensional features such as location, online duration, and visitor's query term [29]. We then explore the behavioral patterns in each interactive session and categorize the users into different groups using the K-means clustering algorithm.

3.1. Geographic Distribution of User. In Figure 1, deeper color means a larger number of users. Through the terminal IP analysis of the province to which the user belongs, it is seen that the major users are from Guangdong province followed by Zhejiang and Shanghai. The number of habitats in these southeastern coastal cities is much higher than that of the inland cities. In the inland cities, users are highly concentrated in economically developed regions such as Beijing, Hubei, and Sichuan. The overall visitor of this application is mainly distributed in the provinces with high information levels and leading cultural development. Analyzing the geographical distribution of the users can help the decision-maker to evaluate the effect of targeted advertising and formulating the delivery strategies.

3.2. Analysis of User Online Duration. Analyzing the distributions of the frequencies over each operation during a day can improve the quality of experience (QoE) and save operational costs for service providers. Besides, it optimizes the allocation of system resources. For instance, operators can reduce network bandwidth during periods of low workload. We take a day as one periodical unit to count the distribution of requests from the users within one day. As shown in Figure 2, by analyzing the request frequency of all users' login time within the period in days, it is seen that the total number of visitors is low overnight before the beginning of a rapid ascent commencing at 4 a.m. Then, it remains active over the day, reaching its initial peak around 11 : 00 a.m., a tiny peak at 4 : 00 p.m., and an evening high between 8 and 10 p.m. It can be found that there is a peak in the morning, noon, and evening. The active periods are concentrated in (9 : 00 a.m. to 11 : 00 a.m.), (14 : 00 p.m. to 16 : 00 p.m.) and (20 : 00 p.m. to 23 : 00 p.m.), with the number of users in (20 : 00 p.m. to 23 : 00 p.m.) being the most active in the evening. This finding is in line with the "2018-2019 Mini Program Industry Growth Research Report" that indicates the active peak time is from 9 a.m. to 11 a.m. and from 2 p.m. to 4 p.m. The peak of users' activity mainly occurs during the idle period when they go off work or school. Combining with more multidimensional data, analyzing and verifying more thorough users' occupation distribution for users are our future work.

3.3. Age Group Estimation. Understanding the correlation between the customers' demographics (e.g., age and gender) and behavior is essential for marketing operators. Age group estimation plays a key role that prompts companies to target potential customers in the right place at the right time and enhance their service [30]. As shown in Figure 3, the birth

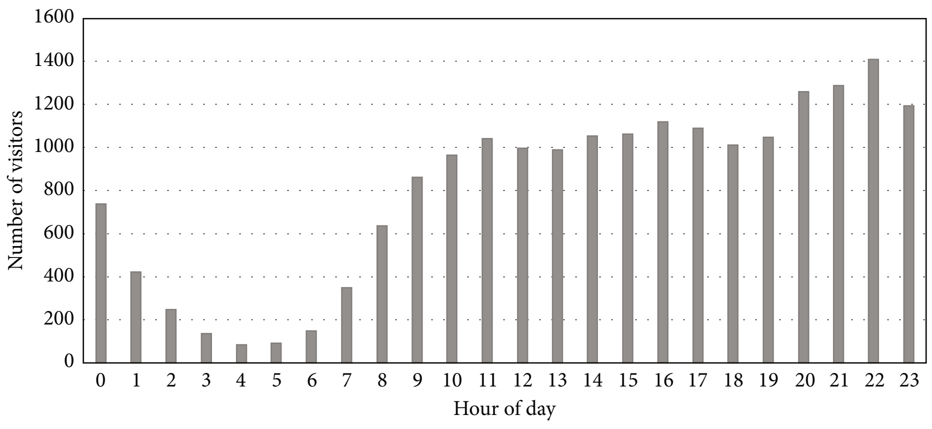


Figure 2: Distribution of the hourly number of online users.

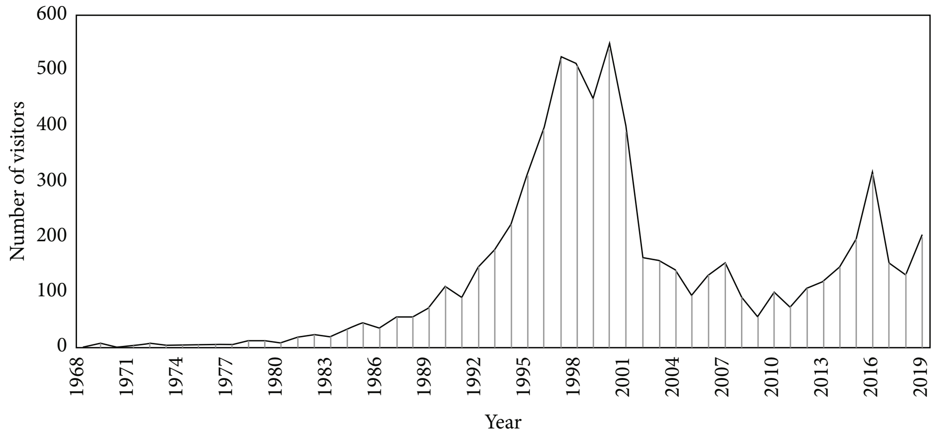


Figure 3: Distribution of the users' birth years.

year selected by users is mostly between the late 1990s and early 2000s. It can be known from the calculation that the age of the user group should be between 18 and 23 years old as of the year 2019, and most of them are university students. Also, we find the number of visitors increases steadily since 2011 until there is a local peak in 2016. For the reason that the population is statistically aged from 3 to 8, we can reasonably assume that they have little ability and accessibility to the Internet and most likely their parents used this application for them.

4. *K*-means Clustering Analysis

In this section, we describe our unsupervised method to build a user behavior model from clickstream log data. Clustering is the process of organizing data into classes that are internally cohesive and well-separated. As a representative of the distance-based unsupervised clustering algorithms, K-means [31] assigns each data point to the cluster which has the closest centroid [32] and produces tighter partitions than the hierarchical clustering. Usually, N samples or observations are divided into K clusters, and the K value is specified by the elbow rule and silhouette score.

Clustering based on original statistical data is feasible in theory. In practice, it has very rarely done so—not only will the redundant and nonbusiness-related features increase the computational complexity, but also such results are not much practical significance.

Users in distinct life stages have different intentions and perform different activities in the process of obtaining information, and the valuable behavioral features within each group are often hidden in the original statistical data. According to the characteristics of clickstream log data and specific business, we construct such indicative features: average interval between user operations(s), average number of operations per session, and average duration of each session(s).

(i) *Average interval between user operations*. It means how much time on interval between two operations in one session

(ii) *Average number of operations per session*. It counts how many times the users interact with this app in one session. The session is defined as a period wherein a user is actively engaged with an app

(iii) *Average duration of each session*. It shows how long users every session last

4.1. Data Normalization. Data normalization is one of the preprocessing in data mining and is especially needed for

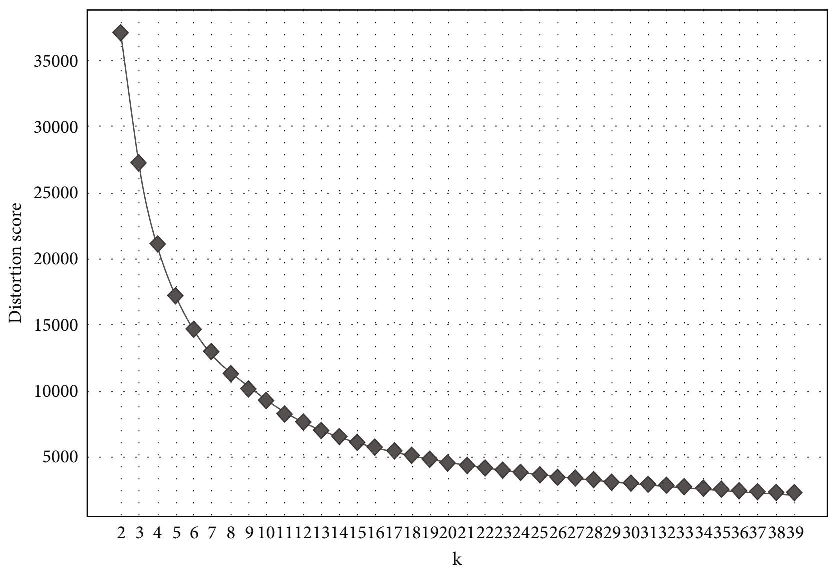


Figure 4: Distortion score of elbow for K-means clustering.

distance metrics, such as Euclidian distance which is sensitive to differences in the magnitude or scales of the features. The distance-based K-means algorithm will give a higher weighting to the variable with higher magnitude, so we use data normalization for all variables to overcome the bias.

The importance of normalization is that it can generate high-quality clusters and improve the performance of clustering algorithms [33]. There are different normalization techniques such as min-max, Z-score, and decimal scaling. In this paper, we adopt Z-score normalization as it handles outliers well, and do not have a predetermined range. The calculation formula is as follows:

$$x_{\text{norm}} = \frac{x - x_{\text{mean}}}{x_{\text{std}}}. \quad (1)$$

4.2. Identify K Value. Before performing K-means clustering on the above data, we need to get the optimal value of K. The most prevalent methods for determining the right cluster numbers are the elbow technique and the silhouette method. The elbow method measures the sum of squared error (SSE) for different K value to choose the most appropriate number of clusters. The SSE will decrease as the K value rises. The lower the SSE, the fewer the samples in each cluster are and the more homogenous they are. We choose K around the point where the degree of distortion lowers the most. The silhouette method estimates the cohesion and the separation [34]. For clustering results, we want modest differences inside clusters and huge disparities across clusters; therefore, the optimal cluster number is determined by the highest score of the index, and the calculation formula is as follows:

$$s(i) = \frac{b(i) - a(i)}{\max\left(a(i), b(i)\right)}, \quad (2)$$

where $b(i)$ represents the mean distance between point i and all sample points in the nearest cluster and $a(i)$ represents the mean distance between point i and other sample points in its cluster.

To identify the optimal K value, we evaluate the performance of the elbow method and the silhouette method on the mobile user log dataset. We use the K-means algorithm with K ranging from 2 to 40. Figures 4 and 5 demonstrate the experimental results. We also consider the specific business on our real-world dataset to be much more realistic and actionable; hence, we subdivide the data sets into 8 groups to keep clustering accuracy. As shown in Figure 4, at $k = 8$, the line graph begins to flatten significantly, and the sum of squared distance (SSE) is 11322.847 when $k = 8$. In Figure 5, the silhouette score is 0.49 at $k = 8$. The structures of clustering results are illustrated in Figure 6. Distinct clusters are clearly shown in 3 dimensions and the clustering algorithm groups data points into nonoverlapping subgroups in a clear and distinct way.

4.3. Analysis of Group Portrait. By qualitatively analyzing the clustering results, we divide the potent visitors into eight groups. The result is consistent with the Pareto principle, generally known as the 80/20 rule. In other words, 20% of users devote 80% of the app's network volume.

Persona 1: As can be seen from the Table 1, this persona contains more operations per session on average, with 77 operations per session, despite the short interval between operations (an average of 12.737 s). From the high number of actions and short intervals, we can presume users may be interested in the main functionalities of this app, but the specific content does not address their genuine demands. The conductor can conduct a questionnaire to capture the essence the users' wants and desires, based on the survey results, taking efforts to improve the content supply to satisfy the needs of diverse users.

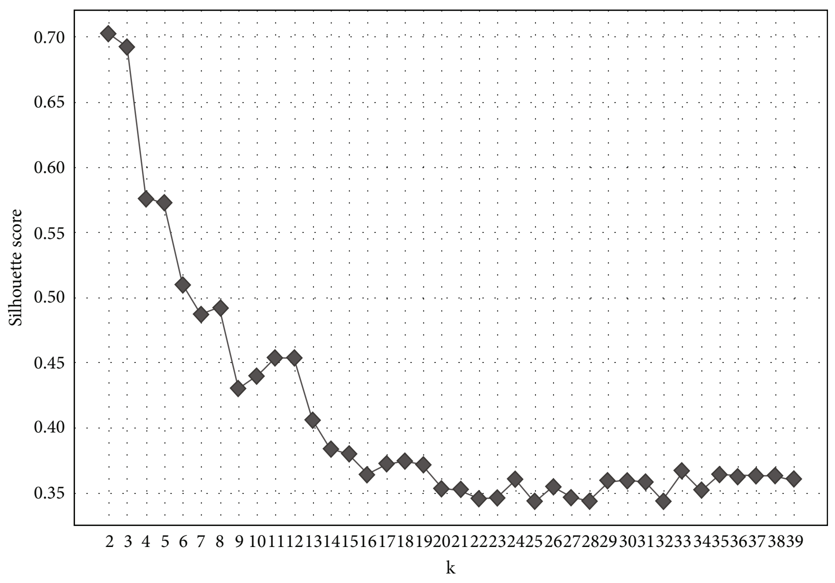


Figure 5: Silhouette score for K-means clustering.

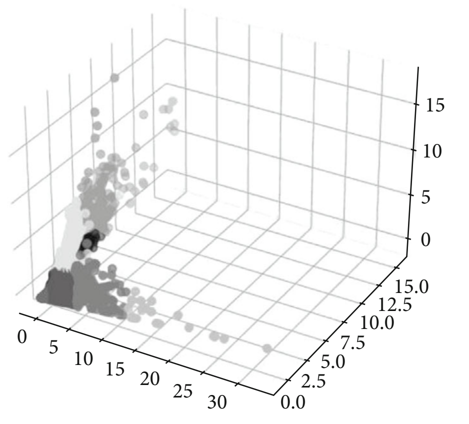


Figure 6: Clustering effect when k is equal to 8.

Persona 2: It accounts for 20.92% of all the analyzed users. The average interval between user actions is 15.071 seconds, with each session having 32 operations on average and lasting 424.124 seconds or nearly 7 minutes. The small number of operations and short duration time imply that this persona does not have a strong desire for continuous usage, which might mean that the content does not meet user demands or that the novice guiding instruction is too complicated. The conductor should perform an investigation to discover the possible causes behind user churn and focus efforts on enhancing capabilities and periodically sending updates to users in order to create a powerful chance to deepen users' comprehension of the product and reestablish bonds with them.

Persona 3: This group of users, on average, spends around 9 minutes in one session, and their average access frequency is 11 times per session. It is assumed that this type of user has an intention to use. To build customer loyalty, the conductor may utilize collaborative recommendations to deliver more information depending on users' interests.

Persona 4: It is the largest cluster with about 65.11% of users with the mean interval between two operations of 14.343 s, each session contains only 9 operations on average, and the mean session duration time is about 1 minute. Based on the statistics, it is reasonable to conclude that this user group does not touch the app's module thoroughly. The conductor can provide eye-catching content to them to pique their attention and encourage the users to return.

Persona 5: Although there are fewer operations in this session, the average session duration is greater, at 18 minutes, and the gap between operations is longer. The session contains fewer operations, but the average session duration is longer, at 18 minutes, and the interval between operations and operations is longer. This signifies that the page's content is popular among users. To raise users' dependency on the product, strengthen their sense of belonging to the platform, and improve user retention rate, the conductor needs completely comprehend their attractive content and perform targeted pushing.

Persona 6: This persona has the most operations in each session (173) and the longest session duration (43 minutes on average). According to the data, this persona has a high level of loyalty and trust in the platform and is the valuable customer who should be the conductor's first focus. Persona 6 is essential for achieving a virtuous cycle of development. To develop a closer relationship with this persona, incentivize consumers with tailored service, points, and unique offers.

Persona 7: This persona has the fewest users, the longest average gap between operations (570.866 s), and the smallest average number of operations each session (just 4). We believe this persona has achieved its objectives with only a few activities. Therefore, for those users, we can employ material incentives to stimulate them to share the product with their friends.

Table 1: The mean value of the behavior features for various users.

Cluster Id	Average interval between user operations (s)	Average number of operations per session	Average duration of each session (s)	Number of users in the cluster	The proportion of users
1	12.737	77	881.229	1024	5.28%
2	15.071	32	423.124	4054	20.92%
3	73.242	11	576.579	887	4.58%
4	14.345	9	100.069	12621	65.11%
5	235.898	8	1087.636	122	0.63%
6	17.497	173	2639.856	181	0.93%
7	570.866	4	957.968	21	0.11%
8	50.088	49	1938.294	473	2.44%

Persona 8: This type of user has a higher access frequency and is interested in the features of the app and willing to take time to indulge in it. For such great potential users, the conductor should focus on delving into the public character of the users and, on that basis, adopt suitable product development strategies to raise users' degree of satisfaction and promote the conversion of such active users into the valuable user.

5. Conclusions

The amount of time spent on using mobile apps (especially for the WeChat Mini Program) has been significantly increased in recent years. User behavior patterns should be thoroughly modeled to provide the operators with deeper insights and expand their targeted customer market. To tackle this important problem, in this work, we visually categorized users with real-world data into different user portrait groups. In particular, we presented a classical clustering analysis quantitatively and qualitatively, where we combine the static user information, e.g., their region, birth year, and gender with their dynamic attributes, such as users' online duration, interaction intensity, and access frequency to find correlations between WeChat Mini Program users. We further compared various internal evaluation measures for the most appropriate clustering results. To the best of our knowledge, this is the first research work to model Mini Program user behavior patterns that provides actionable insights for practitioners. The findings of how to label the targeted mobile users in this paper enhance future products and help companies keep their competitive advantage.

Further research directions include applying more sophisticated algorithms to identify social users from log data; Deng et al. [35] proposed for frequent pattern mining user identification algorithm, extending more comprehensive key attributes to explore user portraits. Consequently, we plot precisely user portraits and then instantly prompt decision-makers to regulate market strategy.

Data Availability

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

Conflicts of Interest

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

Authors' Contributions

Guangmin Li, Wenjing Chen, and Xiaowei Yan contributed equally to this work.

Acknowledgments

This work is partly supported by the Natural Science Foundation of Hubei Province of China (Grant 2020CFB828); the Teaching Research Project of Hubei Normal University (Grant XJ202001); and the Outstanding Young and Middle-aged Technological Innovation Team of Hubei Province of China (Grant T201430).

References

[1] X. Zhou, Y. Xu, Y. Li, A. Josang, and C. Cox, "The state-of-the-art in personalized recommender systems for social networking," *Artificial Intelligence Review*, vol. 37, no. 2, pp. 119–132, 2012.

[2] X. Tao, Y. Li, and N. Zhong, "A personalized ontology model for web information gathering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 4, pp. 496–511, 2011.

[3] S. Kanoje, S. Girase, and D. Mukhopadhyay, "User profiling trends, techniques and applications," https://arxiv.org/abs/1503.07474.

[4] N. N. Ontika, M. F. Kabir, A. Islam, E. Ahmed, and M. N. Huda, "A computational approach to author identification from Bengali song lyrics," in *Proceedings of International Joint Conference on Computational Intelligence, Algorithms for Intelligent Systems*, M. Uddin and J. Bansal, Eds., pp. 359–369, Springer, Singapore, 2020.

[5] L. Zhang, C. Xu, Y. Gao, Y. Han, X. Du, and Z. Tian, "Improved Dota2 lineup recommendation model based on a bidirectional LSTM," *Tsinghua Science and Technology*, vol. 25, no. 6, pp. 712–720, 2020.

[6] H. Xu, H. Zhang, M. Wei, and H. Yin, "Research on the construction of social media user portrait and resource aggregation model," *Library and Information Service*, vol. 63, no. 9, pp. 109–115, 2019.

[7] L. Zhang, X. Zhang, H. Lu, and L. Zhang, "Research on user persona of knowledge online Live's paid-up members," *Library and Information Service*, vol. 63, no. 5, pp. 84–91, 2019.

[8] H. Gu, J. Wang, Z. Wang, B. Zhuang, and F. Su, "Modeling of user portrait through social media," in *2018 IEEE international conference on multimedia and expo (ICME)*, pp. 1–6, San Diego, CA, USA, 2018.

[9] A. Cooper, "The inmates are running the asylum," in *Software-Ergonomie'99*, U. Arend, E. Eberleh, and K. Pitschke, Eds., vol. 53 of Berichte des German Chapter of the ACM, p. 17, Vieweg, Teubner Verlag, Wiesbaden, 1999.

[10] A. Mislove, S. Lehmann, Y.-Y. Ahn, J.-P. Onnela, and J. Rosenquist, "Understanding the demographics of Twitter users," *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 5, no. 1, pp. 554–557, 2011.

[11] P. H. B. Ruas, A. D. Machado, M. C. Silva et al., "Identification and characterisation of Facebook user profiles considering interaction aspects," *Behaviour & Information Technology*, vol. 38, no. 8, pp. 858–872, 2019.

[12] Z. Yu, E. Xu, H. Du, B. Guo, and L. Yao, "Inferring user profile attributes from multidimensional mobile phone sensory data," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5152–5162, 2019.

[13] L. Zhang, Z. Huang, W. Liu, Z. Guo, and Z. Zhang, "Weather radar echo prediction method based on convolution neural network and long short-term memory networks for sustainable e-agriculture," *Journal of Cleaner Production*, vol. 298, article 126776, 2021.

[14] X. Shan, X. Zhang, and X. Liu, "Research on user portrait based on online review: taking Ctrip hotel as an example," *Information Studies: Theory & Application*, vol. 41, no. 4, pp. 99–104 +149, 2018.

[15] H. Zeng and S. Wu, "User image and precision marketing on account of big data in Weibo," *Modern Economic Information*, vol. 16, pp. 306–308, 2016.

[16] L. Liu, S. Wang, and Z. Hu, "A literature review on community profiling," *Library and Information Service*, vol. 63, no. 23, pp. 122–130, 2019.

[17] M. Akbari and T.-S. Chua, "Leveraging behavioral factorization and prior knowledge for community discovery and profiling," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pp. 71–79, Cambridge, United Kingdom, 2017.

[18] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping Multidimensional Data: Recent Advances in Clustering*, J. Kogan, C. Nicholas, and M. Teboulle, Eds., pp. 25–71, Springer, Berlin, Heidelberg, 2006.

[19] J. Blömer, C. Lammersen, M. Schmidt, and C. Sohler, "Theoretical analysis of the k-means algorithm—a survey," in *Algorithm Engineering*, L. Kliemann and P. Sanders, Eds., vol. 9220 of Lecture Notes in Computer Science, , pp. 81–116, Springer, Cham, 2016.

[20] L. Bottou and Y. Bengio, "Convergence properties of the k-means algorithms," *Advances in Neural Information Processing Systems*, vol. 7, 1994.

[21] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[22] X. Wu, V. Kumar, J. Ross Quinlan et al., "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, pp. 1–37, 2008.

[23] S. Pandey, M. Aly, A. Bagherjeiran et al., "Learning to target: what works for behavioral targeting," in *Proceedings of the 20th ACM international conference on Information and knowledge management*, pp. 1805–1814, Glasgow, Scotland, UK, 2011.

[24] R. Saia, L. Boratto, S. Carta, and G. Fenu, "Binary sieves: toward a semantic approach to user segmentation for behavioral targeting," *Future Generation Computer Systems*, vol. 64, pp. 186–197, 2016.

[25] K. Tsiptsis and A. Chorianopoulos, *Data Mining techniques in CRM: inside customer segmentation*, John Wiley & Sons, 2010.

[26] P. Khanthaapha, L. Pipanmaekaporn, and S. Kamonsantiroj, "Topic-based user profile model for POI recommendations," in *Proceedings of the 2nd International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence*, pp. 143–147, Phuket, Thailand, 2018.

[27] S. Mohamed and A. I. Abdelmoty, "Spatio-semantic user profiles in location-based social networks," *International Journal of Data Science and Analytics*, vol. 4, no. 2, pp. 127–142, 2017.

[28] S. Zhao, F. Xu, Z. Luo, S. Li, and G. Pan, "Demographic attributes prediction through app usage behaviors on smartphones," in *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, pp. 870–877, Singapore, Singapore, 2018.

[29] C. Wu, F. Wu, J. Liu, S. He, Y. Huang, and X. Xie, "Neural demographic prediction using search query," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 654–662, Melbourne VIC, Australia, 2019.

[30] Z. Qin, Y. Wang, H. Cheng, Y. Zhou, Z. Sheng, and V. C. M. Leung, "Demographic information prediction: a portrait of smartphone application users," *IEEE Transactions on Emerging Topics in Computing*, vol. 6, no. 3, pp. 432–444, 2018.

[31] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[32] V. R. Patel and R. G. Mehta, "Impact of outlier removal and normalization approach in modified k-means clustering algorithm," *International Journal of Computer Science Issues (IJCSI)*, vol. 8, no. 5, p. 331, 2011.

[33] M. Souto, I. G. Costa, D. Araujo, T. B. Ludermir, and A. Schliep, "Clustering cancer gene expression data: a comparative study," *BMC Bioinformatics*, vol. 9, no. 1, 2008.

[34] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona, "An extensive comparative study of cluster validity indices," *Pattern Recognition*, vol. 46, no. 1, pp. 243–256, 2013.

[35] K. Deng, L. Xing, L. Zheng, H. Wu, P. Xie, and F. Gao, "A user identification algorithm based on user behavior analysis in social networks," *IEEE Access*, vol. 7, pp. 47114–47123, 2019.

Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 1096467, 9 pages
https://doi.org/10.1155/2022/1096467

Research Article

Tunnel Lining Defect Identification Method Based on Small Sample Learning

Anfu Zhu, Congxiao Ma, Shuaihao Chen, Bin Wang, and Heng Guo

North China University of Water Resources and Electric Power, Zhengzhou, China

Correspondence should be addressed to Anfu Zhu; zhuanfu@ncwu.edu.cn

Received 23 June 2022; Accepted 8 August 2022; Published 26 August 2022

Academic Editor: Yawen Chen

Aiming at the problem of insufficient number of samples due to the difficulty of data acquisition in the identification of tunnel lining defects, a generative adversarial network was introduced to expand the data, and the network was improved for the mode collapse problem of the traditional generative adversarial network and the problem that the generated image features were not obvious. On the basis of the WGAN-GP network, a deep convolutional network is selected as its backbone network, and the effectiveness of the deep convolutional network in feature extraction by Lv et al. (2022) is used to improve the quality of the images generated by the network. In addition, the residual module is introduced into the discriminator network, and the upsampling module is introduced into the generator network, which further solves the problem of gradient disappearance of the two networks during the update iteration process through the idea of cross-connection, while better retaining the underlying features, which effectively solves the problem of mode collapse and low quality of generated images in the generative adversarial network. Compared with the original network, the image quality of the generated adversarial network is improved, and the discriminator and generator losses converge faster. At the same time, the recognition accuracy of the YOLOv5 network is improved by 4.4% and the overfitting phenomenon is alleviated, which proves the effectiveness of the method under the limited training data set.

1. Introduction

In today's era, rail transit has become an important means to solve urban traffic jams. Taking advantage of tunnels can effectively avoid the time cost and safety risks caused by traffic jams. However, the existence of the tunnel itself also has many security risks, such as the construction at the beginning of the engineering hidden trouble, time erosion under the aging problem, and the sudden geological disaster caused by the structural hidden trouble. How to avoid and repair these hidden dangers effectively is an important problem for national property and people's safety. Using GPR image combined with deep learning to identify defects in tunnel lining can effectively save manpower and material resources, which is of great significance to the society [1]. However, it is not easy to obtain the data of tunnel lining disease, and there is no public data set for use. The lack of data set is likely to lead to the problem of insufficient precision or poor robustness of the trained identification network. Therefore, how to effectively expand the number of samples is a problem that must be faced.

In the traditional way of expanding data set, the number of samples can be expanded by rotating, clipping, and adding noise to the original data in the way of data enhancement [2]. Goodfellow et al. [3] proposed a generative adversarial network (GAN) model in 2014. Based on the idea of game theory, they constructed a network of mutually opposing generators and discriminators to generate images, and the image generation effect was relatively good. At present, GAN has been successfully applied in image repair and restoration, animation generation, superresolution image reconstruction, and other fields [4]. Arjovsky et al. [5] proposed to use Wasserstein distance to replace KL divergence and JS divergence in the original GAN objective function to construct a WGAN network in view of problems such as training instability and gradient disappearance in the original GAN. However, although WGAN network solves the problem of original GAN in principle, its effect is not significantly improved in the process of image generation compared with traditional GAN. Therefore, Gulrajani et al. [6] proposed Wasserstein generative adversarial network (WGAN-GP) with the introduction of gradient penalty terms. The existence of gradient penalty terms effectively stabilized the training of WGAN network and significantly improved the quality of generated images.

At the application level, Wang Jianlin et al. made use of the effectiveness of WGAN-GP in image generation to solve the problem of low detection accuracy of cooperative target caused by complex component structure and change of measurement environment in 3d precision measurement of large components by expanding the sample number of cooperative target image and realized multitype cooperative target detection [7]. Li et al. used WGAN-GP network to generate rice disease image samples, expanded the small sample set of rice disease image, and effectively enhanced the model training and learning effect [8]. Xu et al. [9] proposed an oversampling model based on convergent WGAN, called convergent WGAN (CWGAN), in order to improve the training stability of GAN oversampling to detect network threats. The training process of CWGAN consists of multiple iterations. In each iteration, the training time of the discriminator is dynamic, depending on the convergence of the discriminator loss function in the last two iterations. When the discriminator is trained to converge, the generator is trained to generate new minority samples.

The above methods still have the problems of poor sample quality or slow convergence of network loss. Aiming at this problem, this paper proposes a WGAN-GP generative adversarial network model based on deep convolutional networks and uses the superiority of deep convolutional networks in feature extraction to improve network performance. At the same time, residual module and upsampling module are introduced into discriminator network and generator network, respectively, which can avoid the loss of bottom features by crossing connections, alleviate the problem of disappearing gradient of network, and improve the stability of network. Finally, the generated defect data is combined with the original data to form an enhanced data set to improve the accuracy and robustness of the recognition network.

2. Introduction to GAN

2.1. Generative Adversarial Networks. Generative adversarial network (GAN) is a generative model proposed by Goodfellow et al. In 2014, its main idea is to make two neural networks continuously play binary minimax game, during which the model gradually learns the real sample distribution. Generally speaking, when the confrontation between two networks reaches Nash equilibrium, the training is considered to be completed [10].

Figure 1 shows the basic model of GAN. The input of generator network (denoted as G) is random variable (denoted as Z) from hidden space (denoted as p_z), and the output generates samples. Its training objective is to improve the similarity

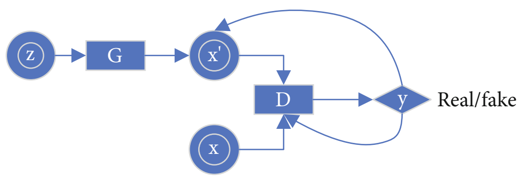


Figure 1: Generate adversarial network model.

between the generated samples and real samples, so that they cannot be distinguished by discriminator network (denoted as D). Let the distribution of the generated sample (called p_g) be as similar as possible to that of the real sample (called p_r). The input of D is the real sample (denoted as x) or the generated sample (denoted as x'), and the discriminant result is output. The training objective is to distinguish the real sample from the generated sample. The discriminant results are used to calculate the objective function, and the network weights are updated by back propagation.

Therefore, the training purpose of GAN can be described as minimizing the distance between p_g and p_r while maximizing the sample discrimination accuracy of discriminator D. Thus, the expression of the objective function can be obtained:

$$\min_G \max_D V(D, G) = E_{x\sim p_r}[\log D(x)] + E_{x'\sim p_g}\left[\log\left(1 - D\left(x'\right)\right)\right]. \tag{1}$$

2.2. WGAN-GP. The problems and challenges that the original GAN has been facing can be summarized in one sentence: the better the discriminator, the more serious the generator gradient disappears. During the training process, if the discriminator is trained too well, the generator will have a problem that it cannot continue to learn. When the discriminator is the optimal discriminator, the optimal discriminator is brought into the generator loss function, and after transformation, we can get the following:

$$E_{x\sim p_{r(x)}}\left[\log \frac{p_{r(x)}}{1/2\left[p_{r(x)} + p_{g(x)}\right]}\right] + E_{x\sim p_{g(x)}}\left[\log \frac{p_{g(x)}}{1/2\left[p_{r(x)} + p_{g(x)}\right]}\right] - 2\log 2. \tag{2}$$

At this point, KL divergence is introduced:

$$\mathrm{KL}(P_1 \| P_2) = E_{x\sim p_1} \log \frac{P_1}{P_2}, \tag{3}$$

and JS divergence is as follows:

$$\mathrm{JS}(P_1 \| P_2) = \frac{1}{2}\mathrm{KL}\left(P_1 \middle\| \frac{P_1 + P_2}{2}\right) + \frac{1}{2}\mathrm{KL}\left(P_2 \middle\| \frac{P_1 + P_2}{2}\right), \tag{4}$$

to measure the similarity between two data distributions, so Equation (2) can be written as follows:

$$2\mathrm{JS}\left(P_r \| P_g\right) - 2\log 2. \tag{5}$$

If two distributions have no overlap at all, or if their overlap is negligible, their JS divergence is a constant log 2. Whether P_r and P_g are far away or close at hand, JS divergence is constant log 2, and this leads to the problem of gradient disappearance. At this point, the generator will not be able to propagate back, resulting in no further learning.

In view of the above problems, Wasserstein distance proposed in literature [5, 11] is used as a new similarity measurement standard. Wasserstein distance, also known as Earth-Mover (EM) distance, is defined as follows:

$$W\left(P_r, P_g\right) = {}^{\inf}_{\gamma\sim\Pi(Pr,Pg)} E_{(x,y)\sim\gamma}[\|x-y\|]. \tag{6}$$

$\Pi(P_r, P_g)$ is the set of all possible joint distributions of p_r and p_g combined. For each possible joint distribution γ, it is possible to sample $(x, y) \sim \gamma$ from it to obtain a true sample x and a generated sample y and calculate the distance between the pair of samples: $\|x-y\|$. $E_{(x,y)\sim\gamma}[\|x-y\|]$ represents the expected distance of samples under the joint distribution γ. And the Wasserstein distance is the minimum expected value of all possible joint distributions. The advantage of Wasserstein distance is that even if two distributions do not overlap, it still reflects the distance between them. The ${}^{\inf}_{\gamma\sim\Pi(Pr,Pg)}$ in Wasserstein distance definition cannot be solved directly, so the author transforms it into the following form based on existing theorems. Then, calculate the expected value of the sample distance under the joint distribution γ. And the lower bound that we can take on this expectation value in all possible joint distributions is defined as the Wasserstein distance. The advantage of Wasserstein distance compared with KL divergence and JS divergence lies in that Wasserstein distance can still reflect their distances even if the two distributions do not overlap. The ${}^{\inf}_{\gamma\sim\Pi(Pr,Pg)}$ of Wasserstein's distance definition cannot be solved directly. The author uses an existing theorem to transform it into the following form:

$$W\left(P_r, P_g\right) = \frac{1}{K} {}^{\sup}_{\|f\|_L\le K} E_{x\sim p_r}[f(x)] - E_{x\sim p_g}[f(x)], \tag{7}$$

where x represents the Lipschitz constant, which is defined as an additional constraint is imposed on a continuous function f, requiring that there exists a constant $K \ge 0$ such that any two elements x_1 and x_2 in the domain are satisfied:

$$|f(x_1) - f(x_2)| \le K\,|x_1 - x_2|. \tag{8}$$

However, although WGAN is proved to be perfect theoretically, the real effect is not very good, mainly because of the Lipschitz continuity condition, and WGAN-GP is an improvement on the Lipschitz continuity condition. Its algorithm process includes two important processes:

(1) Use random numbers to make an interpolation between the generated data (denoted as x') and the real data (denoted as x)

$$\hat{x} \longleftarrow \epsilon x + (1-\epsilon)x'. \tag{9}$$

(2) Introduce gradient penalty term

$$L^{(i)} \longleftarrow D_\omega\left(x'\right) - D_\omega(x) + \lambda\left(\|\nabla_{\hat{x}} D_\omega(\hat{x})\|_2 - 1\right)^2. \tag{10}$$

Advantages of WGAN-GP compared with WGAN are as follows:

(1) When the weight of WGAN is cut (for example, when the weight is cut to [-0.01,+0.01], it will lead to uneven weight distribution), WGAN-GP can make good weight distribution by using gradient punishment and give full play to the learning power of neural network

(2) The gradient of D is the entire sample space including the generated image and the real image. It is very difficult to find the gradient directly. However, using random numbers to make an interpolation between the generated data and the real data, to replace all of the parts can achieve similar results, and it is much simpler

3. Improved WGAN-GP Network Model

3.1. Generator Network Structure. The generator structure model is shown in Figure 2. Compared with the original network, the generator network has made the following improvements:

(1) Upsampling is carried out by transpose convolution

(2) Batch-normalization has been used in each layer except for the input layer of the network to stabilize learning

(3) Remove the full connection layer and use the full convolutional network to increase the stability of the model

(4) The activation function in the network uses ReLU function, and the last layer uses Tanh activation function

In each layer of the module, two different methods are used for upsampling, including one nearest neighbor upsampling and deconvolution. The two results are added and averaged to get the final output. This module can effectively prevent information loss and error caused by upsampling. Different sampling methods can extract different shallow local image features, which is conducive to the extraction of texture features and obtain better generation effect. At the same time, as a shortcut, it is completed directly by simple identity mapping, so there is no need to introduce additional parameters, reducing the computational burden.

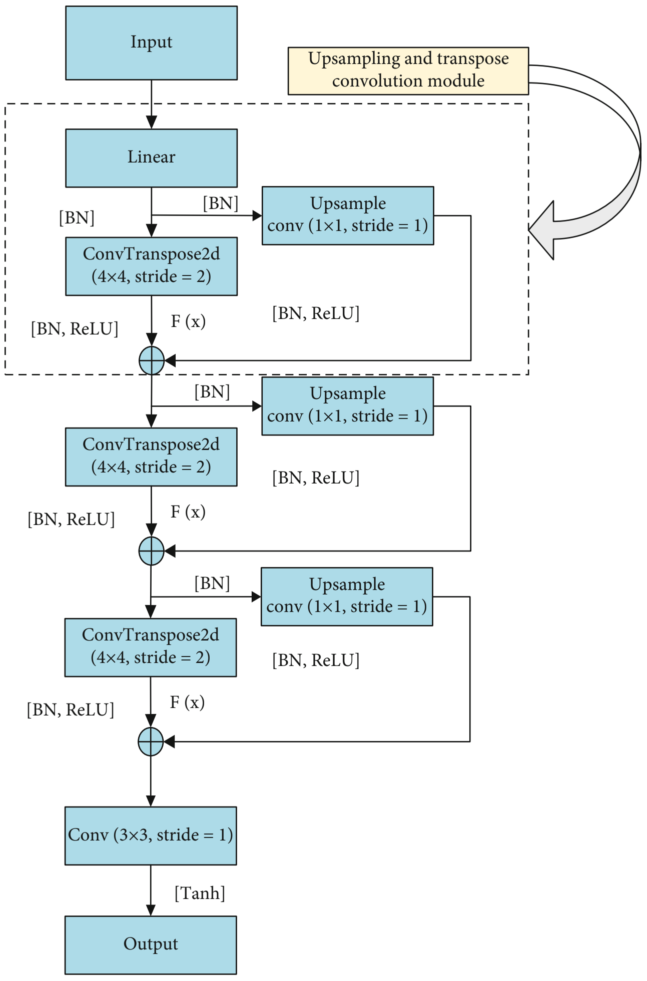

Input
Upsampling and transpose convolution module
Linear
[BN]
[BN]
Upsample conv (1×1, stride = 1)
ConvTranspose2d (4×4, stride = 2)
[BN, ReLU]
F (x)
[BN, ReLU]
[BN]
Upsample conv (1×1, stride = 1)
ConvTranspose2d (4×4, stride = 2)
[BN, ReLU]
[BN, ReLU]
F (x)
[BN]
Upsample conv (1×1, stride = 1)
ConvTranspose2d (4×4, stride = 2)
[BN, ReLU]
[BN, ReLU]
F (x)
Conv (3×3, stride = 1)
[Tanh]
Output

FIGURE 2: Generator structure model.

3.2. Discriminator Network Structure. As the depth of deep network increases, the training of deep learning network becomes more and more difficult, mainly because in the training process of network based on stochastic gradient descent [12], the multilayer back propagation of error signal is very easy to cause the phenomenon of “gradient dispersion” or “gradient explosion.” This phenomenon has troubled the design, training, and application of deeper convolutional neural networks for a long time.

In 2015, the 152-layer ResNet (residual network) proposed by He et al. [13] won the image recognition champion of ILSVRC competition (top1 error 3.6%), which well solved the problem of training difficulties caused by network depth, and its network performance was far superior to traditional network models. Its main idea is to reduce the computational burden through the identity mapping of shortcut (shortcut connection), and its basic residual module network structure is shown in Figure 3.

Assuming that the expected mapping is, and the residual module converts the expected mapping to $H(x) = F(x) + x$ by using the shortcut, then there is $H'(x) = F'(x) + 1$ during back propagation. The existence of constant 1 effectively ensures that the gradient will not disappear when calculating the gradient. At the same time, when the network is propagating, we can ensure that there is more than one path for the information at the bottom layer to be transmitted to the next layer. Even if the gradient of a certain network layer becomes 0, the information will be transmitted across the connection at this time to ensure that the performance of the residual network is better than that of the ordinary network. Therefore, the discriminator network structure of this design is as follows:

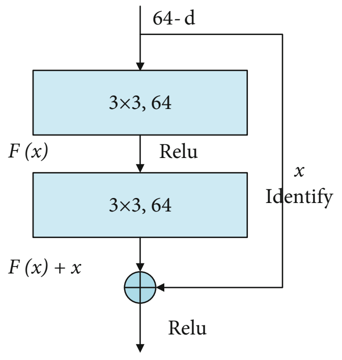


FIGURE 3: Residual module structure diagram.

(1) The convolution of stride and padding was added to replace the pooling layer

(2) In addition to the last output layer, the full connection layer is removed and the convolutional network is used to increase the stability of the model

(3) The LeakyReLU function is selected as the activation function in the network, while the last layer does not use the activation function

(4) Introduce residual modules to optimize the network between each layer

Its structural model is shown in Figure 4.

4. Experiments

4.1. Data Set Establishment. The data used for deep learning this time came from a tunnel bureau in Xi’an. The interior of the tunnel lining was scanned and imaged by ground penetrating radar, and five kinds of disease images, such as voids and incompactness, were selected as input data for model training. A certain proportion is divided into training set and test set. After the traditional data enhancement of the training set and test set, the training set is divided into training set and verification set, which is convenient to understand the training degree in the training process. The collated data set of tunnel lining diseases includes 3800 ground penetrating radar images, and part of the collected disease images are shown in Figure 5.

4.2. Experiment Configuration. In the process of neural network construction, in order to reduce the repeated creation of various tedious environmental codes, the deep learning framework constructed in advance can be used. The deep learning framework used in this experiment is PyTorch framework, which has three main characteristics: concise, high-speed, and easy to use.

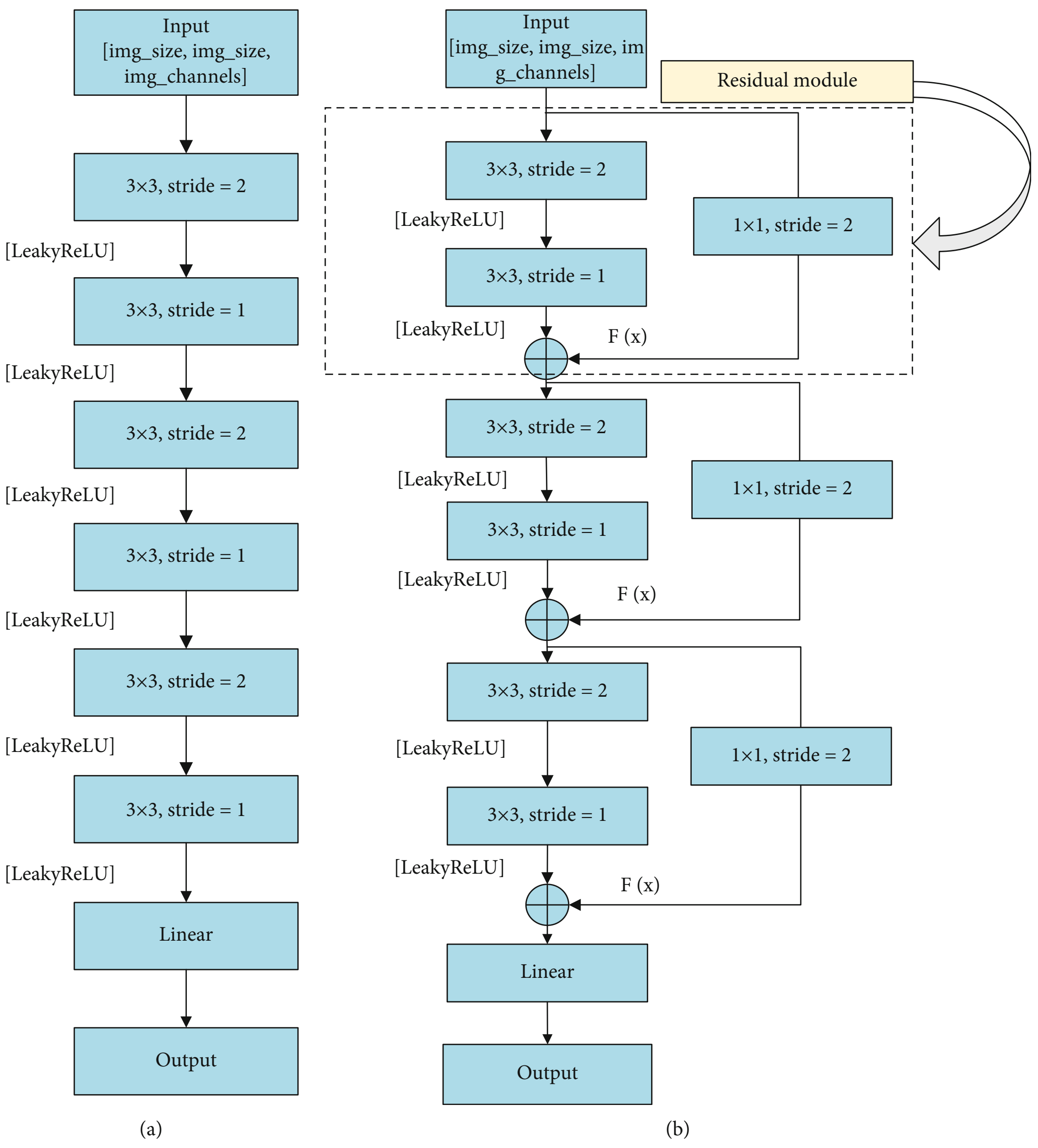


Figure 4: Discriminator network structure diagram: (a) no residual module network and (b) discriminator residual network model.

PyTorch is designed for minimal encapsulation, using existing code rather than rewriting existing code. PyTorch's simplicity allows developers to understand the logic of each step of the code. At the same time, the code of PyTorch framework runs faster than many frameworks, and the implementation and invocation of the code is very convenient, so this framework is selected as the construction framework of all the networks in this experiment.

The experimental environment of this paper is Windows10 system, in which the batchsize of the improved WGAN-GP network model is set as 64, and that of YOLOv5 network is set as 2. The improved WGN-GP network uses Adam optimizer, and the initial learning rate is set to 0.0002 with 3000 iterations. YOLOv5 network iterates 200 times. Table 1 lists the detailed parameters of the experimental environment.

4.3. Result Analysis

(1) After the improved WGAN-GP network is defined, the collected data set of tunnel lining diseases is used to train the improved WGAN-GP network and compare it with the original network. As shown in Figure 6, the feature of disease image generated by WGAN-GP network reconstructed by convolutional network is significantly better than that generated by WGAN-GP network constructed by full-connection layer. The improved WGAN-GP network based on convolutional layer is obviously clearer than the WGAN-GP network based on convolutional layer. In addition, the number of parameters in the original WGAN-GP network using full connection layer is

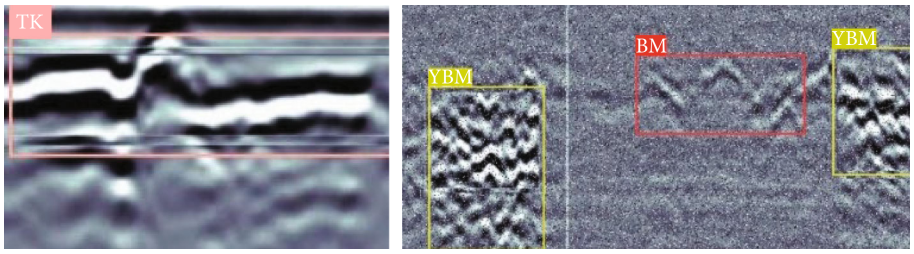


Figure 5: Tunnel disease images: "TK" stands for "void beneath"; "BM" stands for "not dense"; "YBM" stands for "seriously not dense."

Table 1: The software and hardware environment of the experiment.

Configuration	Content
The operating system	Windows10
The graphics card	Nvidia GeForce GTX 3070TI
Memory	8G
CPU	12th Gen Intel(R) Core(TM) i5-12400F
Programming language	Python
Deep learning framework	PyTorch (version 1.11)
Programming platform	Pycharm

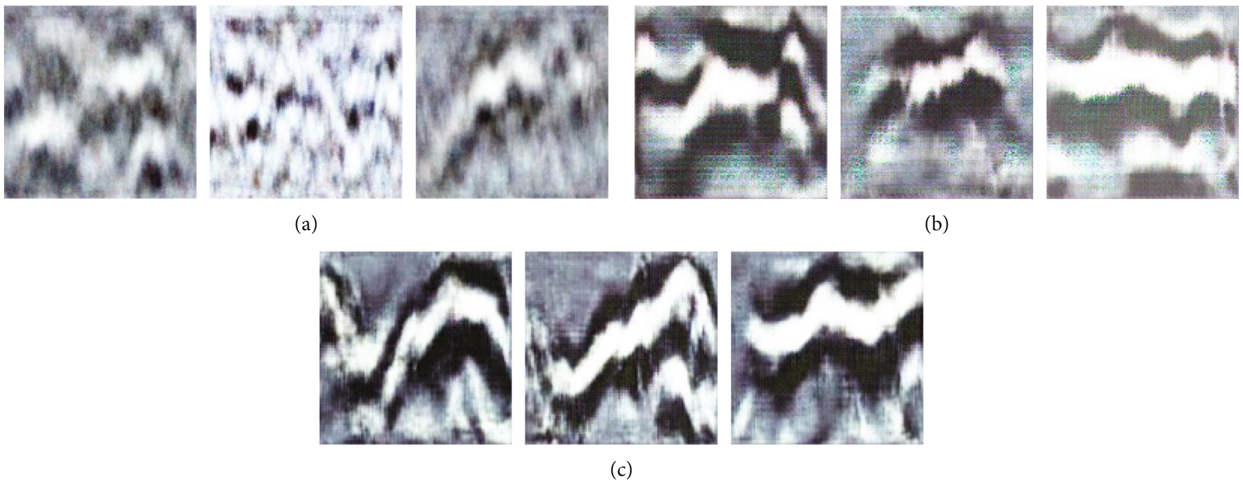


Figure 6: Comparison of image quality generated by WGAN-GP networks with different structures (Epoch = 500, Imgsize = 64 ∗ 64, and label: empty). (a) WGAN-GP (full-connection layer structure) network generates disease samples. (b) WGAN-GP (deep convolution layer structure) network generates disease samples. (c) WGAN-GP (shortcut structure) network generates disease samples.

25.2 m, and the number of parameters in the improved network is only 1.77 m, which reduces 92.98%, greatly reducing the operation cost

(2)At the same time, compared with the generation network that does not introduce the residual module in the discriminator, the improved generative confrontation network converges faster in both D_loss and G_loss and can be generated earlier. For stable and high-quality disease images, the loss functions of the two networks are shown in figure 7

(3)In the official code given by YOLOv5 [14], its target detection network has four versions of different depth and breadth, namely, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x models. Among them, YOLOv5s has the smallest depth and the smallest width of feature map, while YOLOv5l is the largest. Take YOLOv5s as an example, its network structure is shown in Figure 8, which can be divided into input, backbone, neck, and prediction according to different stages of data processing. Among them, data enhancement, adaptive picture scaling, and Ancho box calculation of input image data are all completed in the input part. Backbone uses focus structure and CSP structure to extract the characteristic information of input data and sends it to neck for further study. The neck part adopts FPN+PAN [15, 16]

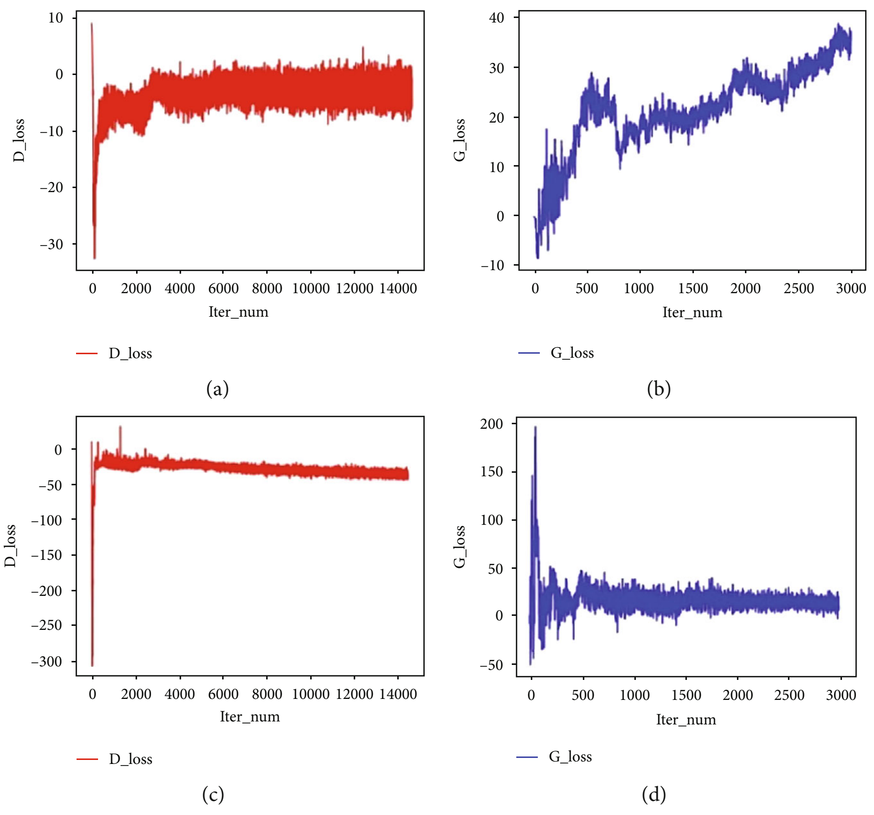


Figure 7: Comparison of convergence speed of network loss: (a) no residual module discriminator loss; (b) no residual module discriminator loss; (c) including residual module discriminator loss; (d) including residual module discriminator loss.

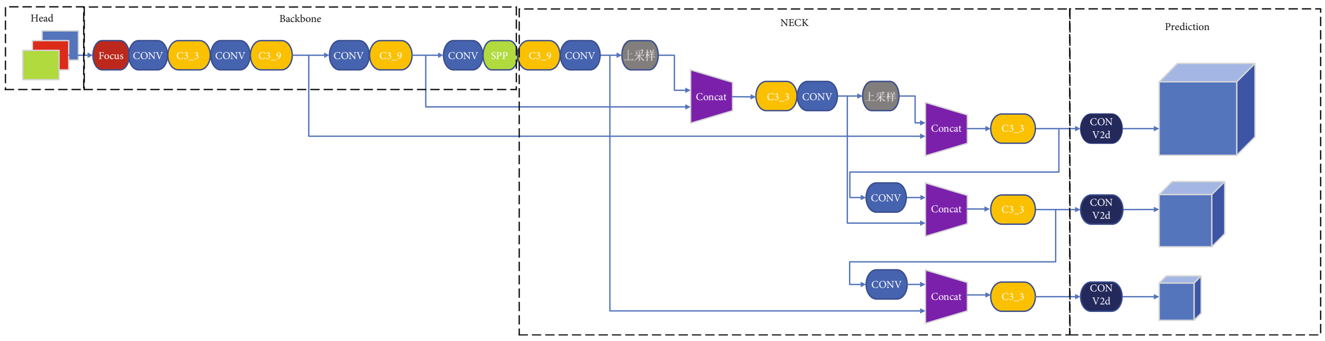


Figure 8: YOLOv5s network structure diagram.

Table 2: Identification accuracy table of tunnel defect data (excluding generated pictures).

Defect types	Val.mAP@.5	Test.mAP@.5	Label
Not dense	98.5%	90.4%	0
Void beneath	98.9%	86.8%	1
Cavity	98.5%	88.7%	2
Seepage	99.5%	77.6%	3
Seriously not dense	98.0%	70.3%	4
All	98.7%	82.8%	All

Table 3: Identification accuracy table of tunnel defect data (include generated images).

Defect types	Val.mAP@.5	Test.mAP@.5	Label
Not dense	98.0%	91.4%	0
Void beneath	99.0%	88.2%	1
Cavity	98.9%	89.3%	2
Seepage	97.6%	79.4%	3
Seriously not dense	97.8%	87.6%	4
All	98.3%	87.2%	All

structure, and the CSP2 structure is adopted to strengthen the ability of network feature fusion. Finally, prediction is made in the prediction part, loss function is calculated, and NMS [17] (nonmaximum suppression) is used to screen the multiobjective box

Finally, 1909 images were selected for the data samples to be integrated and labeled, and then, the original data set was mixed and sent to YOLOv5 network for training and recognition. Under the same test set, the tunnel disease test results are shown in Tables 2 and 3.

5. Conclusions

Generative adversarial networks have always been a hot research field in artificial intelligence. Its advantage lies in the introduction of a new data enhancement method, which can greatly improve the robustness and security of neural networks. However, it also faces prominent problems, including the efficiency of model training, the probability of model crash, and the quality of generated samples. Therefore, it is of great significance to further study and explore generative adversarial networks.

Under the premise of limited tunnel image disease data, this paper introduces generative adversarial network to expand the data and makes improvements to the problems existing in the original generative adversarial network, which greatly improves the quality of generated samples. At the same time, compared with the original data set, the average recognition rate of the training network using the expanded data set is increased by 4.4%, and the recognition rate of all classes is greatly improved. The experimental results show that the improved WGAN-GP network designed in this paper effectively improves the quality of generated samples, strengthens the stability of generative adversace network, and improves the recognition accuracy of the network effectively with the expanded data samples.

In this experiment, the generative adversarial network is mainly improved. In future research, in addition to further optimizing the generative adversarial network, the used recognition network can also be improved, which is believed to improve the recognition rate of diseases.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no competing interest.

Acknowledgments

This study is funded by the Key Scientific Research Project of Universities in Henan Province (19A510004) and the Key Science and Technology Project of Henan Province (222102210135).

References

[1] L. Zhang, Z. Huang, W. Liu, Z. Guo, and Z. Zhang, “Weather radar echo prediction method based on convolution neural network and long short-term memory networks for sustainable e-agriculture,” *Journal of Cleaner Production*, vol. 298, p. 126776, 2021.

[2] C. Shorten and T. M. Khoshgoftaar, “A survey on image data augmentation for deep learning,” *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza et al., “Generative adversarial nets,” *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[4] Y. Zhao, G. Fu, H. Wang, S. Zhang, and M. Yue, “Generative adversarial network-based edge-preserving superresolution reconstruction of infrared images,” *International Journal of Digital Multimedia Broadcasting*, vol. 2021, Article ID 5519508, 12 pages, 2021.

[5] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein generative adversarial networks,” *International Conference on Machine Learning*, vol. 70, pp. 214–223, 2017.

[6] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of wasserstein gans,” *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[7] W. Jianlin, X. S. Fu, Z. C. Huang, Y. Q. Guo, R. T. Wang, and L. Q. Zhao, “multi-type cooperative targets detection using improved YOLOv2 convolutional neural network,” *Optics and Precision Engineering*, vol. 28, no. 1, pp. 251–260, 2020.

[8] L. Jianning, L. Yang, T. Xianpeng, L. Liyuan, and S. Tong, “Research on rice disease image generation method based on WGAN-GP,” *Information Recording Materials*, vol. 22, no. 8, pp. 235–238, 2021.

[9] Y. Xu, X. Zhang, Z. Qiu, X. Zhang, J. Qiu, and H. Zhang, “Oversampling imbalanced data based on convergent WGAN for network threat detection,” *Security and Communication Networks*, vol. 2021, Article ID 9206440, 14 pages, 2021.

[10] W. Zhenglong and Z. Baowen, “Survey of generative adversarial network,” *Chinese Journal of Network and Information Security*, vol. 7, no. 4, pp. 68–85, 2021.

[11] M. Arjovsky and L. Bottou, “Towards principled methods for training generative adversarial networks,” 2017, http://arxiv.org/abs/1701.04862.

[12] L. Lv, J. Chen, L. Zhang, and F. Zhang, “Gradient-based neural networks for solving periodic Sylvester matrix equations,” *Journal of the Franklin Institute*, vol. 600, 2022.

[13] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, Las Vegas, NV, USA, 2016.

[14] D. Thuan, *Evolution of Yolo Algorithm and yolov5: The State-of-the-Art Object Detection Algorithm*, theseus, 2021.

[15] Y. Gong, X. Yu, Y. Ding, X. Peng, J. Zhao, and Z. Han, “Effective fusion factor in FPN for tiny object detection,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021.

[16] W. Wang, E. Xie, X. Song et al., “Efficient and accurate arbitrary-shaped text detection with pixel aggregation network,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, Korea, 2019.

[17] M. Gong, D. Wang, X. Zhao, H. Guo, D. Luo, and M. Song, “A review of non-maximum suppression algorithms for deep learning target detection,” *Seventh Symposium on Novel Photoelectronic Detection Technology and Applications*, vol. 11763, pp. 821–828, 2021.

Hindawi
Wireless Communications and Mobile Computing
Volume 2023, Article ID 9785959, 1 page
https://doi.org/10.1155/2023/9785959

Retraction

Retracted: Intelligent Application of Raw Material Supply Chain Planning System Based on Genetic Algorithm

Wireless Communications and Mobile Computing

Received 17 October 2023; Accepted 17 October 2023; Published 18 October 2023

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope
(2) Discrepancies in the description of the research reported
(3) Discrepancies between the availability of data and the research described
(4) Inappropriate citations
(5) Incoherent, meaningless and/or irrelevant content included in the article
(6) Peer-review manipulation

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] F. Xu, Y. Zhang, Y. Su, J. Li, and J. Zhu, "Intelligent Application of Raw Material Supply Chain Planning System Based on Genetic Algorithm," *Wireless Communications and Mobile Computing*, vol. 2022, Article ID 5054529, 13 pages, 2022.

Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 5054529, 13 pages
https://doi.org/10.1155/2022/5054529

Research Article

Intelligent Application of Raw Material Supply Chain Planning System Based on Genetic Algorithm

Feng Xu,[1] Yu-Meng Zhang,[2] Yi Su,[2] Jia Li,[3] and Jia-Ming Zhu[1]

[1]*School of Statistics and Applied Mathematics, Anhui University of Finance and Economics, Bengbu 233030, China*
[2]*School of Finance, Anhui University of Finance and Economics, Bengbu 233030, China*
[3]*Department of Accounting and Finance, Nanjing Agricultural University, Nanjing 210095, China*

Correspondence should be addressed to Jia-Ming Zhu; zhujm1973@163.com

Received 29 April 2022; Accepted 13 July 2022; Published 23 July 2022

Academic Editor: Lingling Lv

For the ordering and transportation of raw materials of production enterprises, an intelligent supply chain planning system based on genetic algorithms has been researched and developed. Based on the consideration of multicycle and multiraw materials, by constructing a multiobjective function, taking into account the optimal supplier, the optimal economic ordering scheme, and the minimum loss transshipment scheme, an optimal supply planning model based on genetic algorithm is proposed, and the optimal supply chain combination of the next 24 cycles is predicted by using the model, and its applicability is verified. The study shows that the supply chain planning system has good operation convenience, fast, intuitive, practical application effect of the function and can adapt to modern intelligent logistics and transportation.

1. Introduction

Under the current world development trend, the competition of a single enterprise has transitioned to the competition between supply chains. For manufacturing enterprises, the ordering research of enterprise raw materials continues to heat up. Faced with the current environmental trends, traditional manufacturing industries should consider how to build a selection and evaluation system for raw materials and suppliers, formulate and optimize the ordering and transportation plans for raw materials before production, and fully realize the optimization of their supply chain system. However, reasonable supply chain pricing and ordering decisions are very difficult and require long-term unremitting efforts.

In view of the current situation of the ordering and transportation of raw materials of production enterprises, this paper studies the following problems: (i) using relevant data to evaluate the supply characteristics of enterprise suppliers; (ii) quantitative analysis of supply characteristics, establishing a mathematical model that reflects the importance of ensuring the production of enterprises, and using data; (iii) taking whether suppliers are selected as a decision-making variable, using the 0-1 planning model with the least number of suppliers as the target function, and solving the choice of suppliers; (iv) according to the most economical enterprise for the expected goal of the ordering program; (v) introduce new indicators to formulate the transfer scheme of the expected state; and (vi) finally construct a multiobjective system, use genetic algorithms to solve the objective function, and obtain a new optimal ordering scheme and transshipment scheme under the agreed conditions.

In this article, we build an intelligent supply chain planning system. And under the premise of ensuring the supply and demand of raw materials of enterprises, and taking into account the optimization of many factors such as upstream and downstream enterprises in the raw material supply chain, such as the most economical ordering scheme and the transhipper scheme with the least loss, a genetic algorithm model is established to optimize the allocation of the supplier team and predict the possible optimal supply chain combination of raw materials in the next 24 weeks. Through the information-based supply chain planning system, the logistics activities, quantity, and speed are effectively adjusted and planned. The research method in this paper has good practicality and innovation in the prediction of optimal supply chain combination of enterprises.

2. Literature Review

In recent years, the adjustment of the global supply chain pattern is accelerating, and the effective coordination of upstream and downstream supporting industries will increase the viscosity of the supply chain. Due to the changes in the overall situation of the world, how to optimize and upgrade the supply chain of traditional manufacturing enterprises has become a topic of research by many scholars. There are already many excellent scholars who have provided models and schemes for the raw material ordering scheme of enterprises according to different influence ordering angles.

Chen et al. [1] proposed an EPQ integration model for spoiled items that considers both the inventory cost of finished products and raw materials in the inventory study of inventory management. The optimal solution of the model is obtained by using the iterative optimization method to obtain the optimal number of raw material orders in the planning period, the optimal production times, and the optimal service level in the raw material ordering cycle. Huo and Xuan [2] mentioned in the article that the demand for products of production enterprises is affected by sales prices, and the size of product demand will inevitably affect the economic batch of raw materials. Peng et al. [3] proposes a rolling procurement strategy based on time series for the procurement of raw materials in the manufacturing process of complex products. The core idea of this strategy is to implement rolling procurement under normal processing conditions. According to the future price fluctuations of raw materials, a multistage procurement model with the goal of minimizing total cost is established to solve the optimal purchase volume of each stage. Sutrisno [4] proposed that raw material sourcing and product mixing planning are two important components of the manufacturer's industry, based on their significant contribution to production costs and profits. It describes a newly developed mathematical model in the form of multiobjective optimization as an alternative method that may be used to improve multi-period raw material procurement and product mixing planning under uncertain demand.

Research on the analysis of raw material supply chains based on genetic algorithms, for example, Wang and Li [5] used Sheffield University's genetic algorithm toolbox to combine genetic algorithms with linear programming algorithms to solve the model. It is also shown that the influence of demand change is greater than the effect of distance change, and the effect of demand stochastic on the optimal cost and the optimal individual is not large. Dai [6] used genetic algorithms to construct a collaborative supply chain network to improve the effectiveness and efficiency of the supply chain network. However, previous studies are relatively simple, most of them study the same cycle supply chain problem, or assume that only one raw material is used to produce products. But in real life, there may be more than one raw material used in the production of products, and the supply chain network is a multicycle process. Therefore, under the premise of considering multicycle and multiraw materials, this paper also considers the optimization of the schemes of forwarders and suppliers and builds an optimization model of the supply chain system and proves that the system has good applicability in reality.

3. Basic Assumptions

To address this issue, we make the following assumptions: (i) that the data we collect on data related to suppliers and forwarders is true and reliable; (ii) that the forwarder defaults to the maximum capacity in terms of the quantity of raw materials to be transported; (iii) that the prices of raw materials in categories A, B, and C remain unchanged during the study period and future forecast periods; and (iv) that the company will continue to use categories A, B, and C by default for the next 24 weeks.

4. Enterprise Upstream Supply Chain Efficient Management Implementation Path Analysis

4.1. Construction of Raw Material Supplier Evaluation System Based on TOPSIS Improved Factor Analysis

4.1.1. Research Thought. First, visualize the data to visually observe the changes in the weekly order quantity of the enterprise and the weekly supply of suppliers; then divide the indicators into 5 indicators: supply capacity (total average supply), supply stability, and effective supply rate, the actual weekly average value of supply and raw material cost, use factor analysis to determine the indicators and their weights [7], establish a supply feature evaluation model, and construct an evaluation system with 5 indicators extracted from the data, and use a factor analysis model based on the TOPSIS distance method to determine factor weights, calculate composite factor scores and scores [8], and rank businesses, as shown in Figure 1.

We can see that the order quantity of the enterprise and the supply quantity of the supplier change periodically, and there is a certain rule. The subsequent model construction problem can be analyzed according to this.

4.1.2. Model Preparation. First, visualize the data of enterprise order quantity and supplier supply quantity. Figure 2 shows the data change chart of the company's order volume, and Figure 3 shows the supply quantity of the supplier change periodically, and there is a certain rule, and the subsequent model construction problem can be analyzed according to this.

Next, taking into account the factors affecting the development of the upstream supply chain of an enterprise, a multilevel evaluation structure should be established to

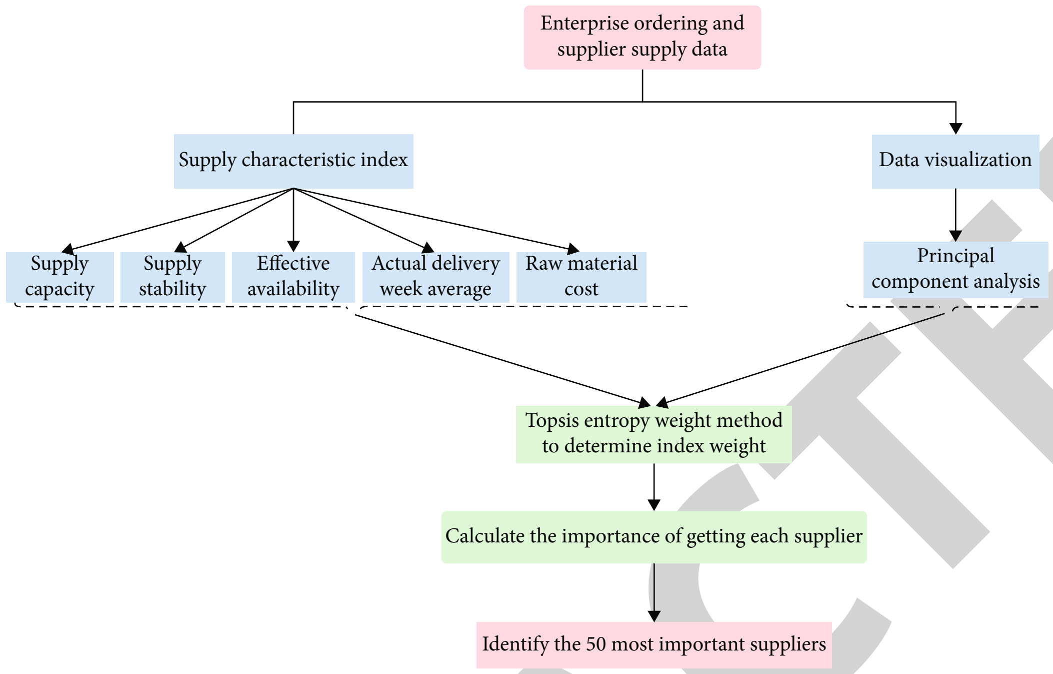

Enterprise ordering and supplier supply data
Supply characteristic index
Data visualization
Supply capacity
Supply stability
Effective availability
Actual delivery week average
Raw material cost
Principal component analysis
Topsis entropy weight method to determine index weight
Calculate the importance of getting each supplier
Identify the 50 most important suppliers

FIGURE 1: Overall thinking process.

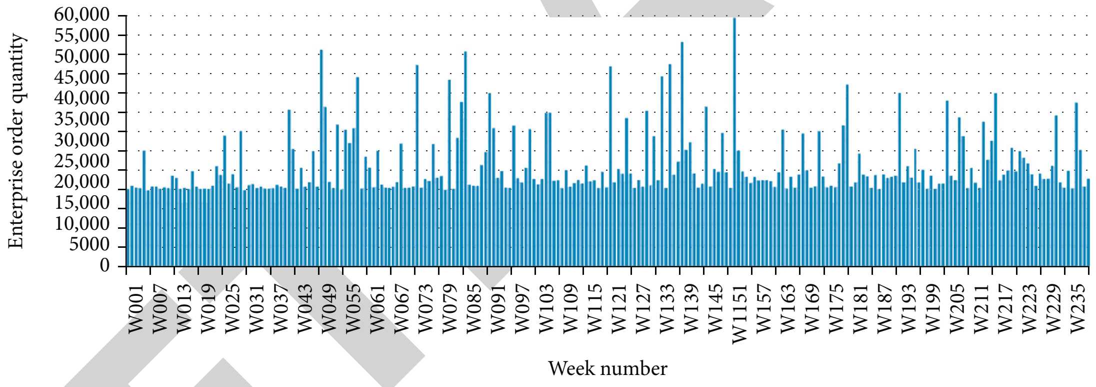


FIGURE 2: The change curve of the order quantity of the enterprise with time.

evaluate raw material suppliers. We mainly analyzed five indicators including supply capacity (total average supply), supply stability, effective supply rate, actual supply week average, and raw material cost. Each aspect corresponds to several specific indicators.

Among them, the supply capacity is the average of the total supply of each supplier in 240 weeks, the supply stability is the sum of the actual supply cycles in 240 weeks, and the actual weekly average is the supply and supply stability of each supplier [9]. The ratio of availability, the effective supply ratio, the ratio of the sum of the number of cycles to the actual order quantity to provide the supply quantity that matches the order quantity or the excess order quantity, and the raw material cost. The weight is [0.75, 0.25], and on this basis, the three materials of ABC are simply quantified.

We use the evaluation indicators established above to establish the evaluation model, taking into account the following:

(1) The stronger the supply capacity (total average supply capacity), the higher the importance of guaranteeing the production of the enterprise

(2) The better the supply stability, the higher the importance of ensuring the production of enterprises

(3) The greater the effective supply rate, the higher the importance of ensuring the production of enterprises

(4) The more the actual weekly average of supply, the higher the importance of guaranteeing the production of enterprises

(5) The lower the raw material cost, the higher the importance of ensuring the production of enterprises

(6) After establishing the index system, we obtained the data of 402 raw material suppliers of the company in the past five years from the official website. Because

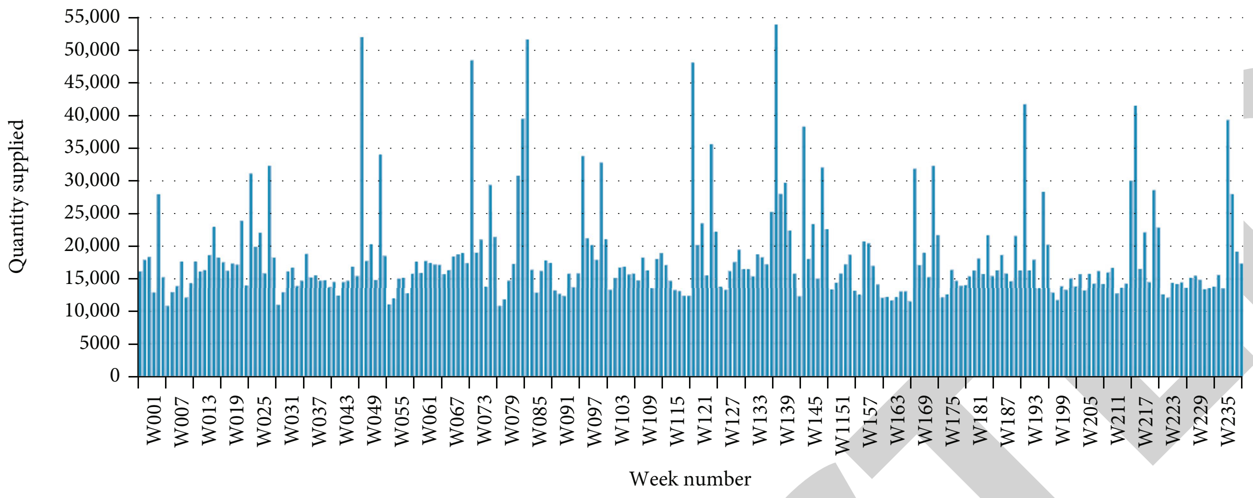


Figure 3: The change curve of the supply quantity of the enterprise with time.

the original data is large and incomplete, it is necessary to preprocess the data [10]

(i) Step 1: according to the existing literature materials, the initial criteria for supplier selection are established: when the supplier's supply volume is 0 for more than one year (48 weeks), it is determined that the transaction viscosity with the enterprise is low, and this type of supply quotient data will be excluded

(ii) Step 2: secondary screening of the data generated by condition (i): when the supplier has a supply volume greater than 80 m^3 for ≧1 week, it is determined that the supplier and the enterprise have a high-intensity transaction relationship under special circumstances, so the supplier will be retained, merchant data.

(iii) Step 3: data normalization and data standardization:

For positive indicators,

$$x_{ij} = \frac{x_{ij} - \min x_{ij}}{\max x_{ij} - \min x_{ij}}. \quad (1)$$

For contrarian indicators,

$$x_{ij} = \frac{\max x_{ij} - x_{ij}}{\max x_{ij} - \min x_{ij}}. \quad (2)$$

The indicator data values are uniformly transformed into the [0, 1] interval. The model uses linear transformation method to standardize the index data [11]. Among them, x_{ij} is the standardized index. In the raw material supplier evaluation system, x_{ij} represents the actual value of the sample data of the jth index of the ith supplier.

4.1.3. Model Establishment and Solution. There are several steps in the process of establishing TOPSIS and the entropy weight model. According to the established index system, SPSS software was used to calculate and solve the model.

(i) KMO and Bartlett's sphericity test: when performing factor analysis, it is necessary to carry out feasibility test on the data and pass the test. The result is credible. The results of the data feasibility test were obtained through SPSS analysis, as shown in Table 1. When the data that is usually suitable for factor analysis is tested, the KMO value is greater than 0.5, which is more suitable for factor analysis [12]. Bartlett's sphericity test corresponds to a P value of 0. 000 ($0.000 < 0.001$), indicating that we reject the null hypothesis at the 99% confidence level, there is a strong correlation between the variables, and the selected indicators are suitable for factor analysis [13]

(ii) Extract the common factor: the common factor variance table can reflect the extracted information of each original variable, and most of the indicators have been extracted about 90% of the information, indicating that most of the information of the original variables has been preserved [14], and the established model can reflect corporate credit risk

As shown in Table 2, the characteristic root and variance contribution rate table, in this paper, according to the two common factors extracted from the gravel diagram, the variance contribution rates are 40.073% and 32.913%, respectively, and the cumulative variance contribution rate reaches 72.986%. Therefore, keeping the first 2 common factors preserves most of the information of the original variable.

(iii) Naming of common factors. According to the common factor, we use the maximum variance rotation

Table 1: KMO and Bartlett's test.

KMO sampling suitability quantity		0.578
Bartlett's sphericity test	Approximate chi-square	279.351
	Degrees of freedom	10
	Salience	≤0.001

method to rotate the factor, simplify the factor structure, and make the actual meaning of the common factor clearer. The rotated composition matrix is shown in Table 3

The first common factor has a larger load on the average of total supply and the average of actual supply weeks, which reflects the quantity of raw materials supplied by suppliers, so it is named as supplier supply. The second common factor is the supply stability, the effective supply rate, and the raw material cost, which has a larger load, reflecting stability and quality of the supplier's supply [15], so it is named the supplier's supply quality factor. The rotation method adopts the Caesar normalization maximum method, expressed as the rotation has converged after 3 iterations.

(iv) Comprehensive factor scores and Rankin

Divide the respective variance contribution rates of the common factors by the cumulative variance contribution rates to calculate the relative variance contribution rates after factor rotation [16]. The relative variance contribution rates of the supplier supply capability factor and the supplier supply quality factor are 66.20% and 66.20%, respectively. 33.80%, which is then assigned as a weight.

According to the component score coefficient matrix, calculate the score function of supplier supply ability factor and supplier supply quality factor:

$$Y_1 = 0.889X_1 + 0.439X_2 + 0.258X_3 + 0.917X_4 - 0.310X_5 \tag{3}$$

$$Y_2 = 0.249X_1 + 0.761X_2 + 0.793X_3 - 0.009X_4 + 0.613X_5. \tag{4}$$

The comprehensive score function is obtained by the calculated relative variance contribution rate:

$$Y = 0.662Y_1 + 0.338Y_2. \tag{5}$$

Solve to get the comprehensive score of each company and rank it, and select the top 15 suppliers are shown as Figure 4.

As shown in Figure 4, we can see that in the 5 indicators of the established raw material supplier evaluation model, in terms of supply capacity and actual supply cycle, there are still deficiencies in the upstream industrial chain of enterprises, and corresponding improvement plans need to be made.

5. Optimal Ordering and Transport Model Based on Multiobjective Optimization and Genetic Algorithm

5.1. Research Thought. We seek the best raw material ordering and transshipment solutions and select the least transfer options on this basis. First of all, we take whether the supplier chooses as the decision variable, takes the minimum number of suppliers as the objective function, establishes a planning model [17, 18], and obtains the minimum number of suppliers supplied; then in the ordering process, the weekly supply of the requested supplier is taken as the decision variable, and the most economical raw material price is taken as the target function; in the transfer process, the supply data is used to obtain the optimal transfer scheme with the lowest transfer loss rate as the target function [19], and finally, we construct a multiobjective function to find the optimal solution. Through the genetic algorithm, the constraints are included in the moderate evaluation, which effectively improves the solution quality, overcomes the disadvantages of local optimization [20], and searches for the optimal solution in the global situation in a faster way, which is suitable for this large-scale optimization problem.

5.2. Model Preparation. Before the model can be established, it is first necessary to design a minimum supplier scheme and find the minimum number of suppliers, that is, each supplier selected should supply as much as possible to the production enterprise [21].

(i) Step 1: the model assumes: first, the loss rate in the transshipment process is taken as 2%, and the receiving volume of the production enterprise is uniformly regarded as 98% of the supply [22]. Second, the maximum supply that each supplier can provide in the past and each cycle point is set as a fixed value, which is expressed as MAX_A, MAX_B, and MAX_C.

$$\text{MAX}_\text{A} = \begin{pmatrix} 2 & 0 & \cdots & 1 \\ 65 & 64 & \cdots & 84 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \tag{6}$$

(ii) Step 2: decision variables. For each supplier, it is only necessary to judge whether to choose or not. Therefore, taking the 146 suppliers of supply A as an example, the selection of 146 suppliers in 24 weeks is made into a matrix of 24 rows and 146 columns S_A as a decision variable, and the value of each position can only be 0 or 1, 1 is expressed as the selection of the supplier, and the purchase is based on the maximum supply [23], or 0 means no purchase. S_B, S_C too

(iii) Step 3: objective function. With the help of a matrix S_A, S_B, S_C, add up the 24-selection data for each column of each house, and if this data is not equal to 0, then the store is selected during the supply process.

Table 2: Characteristic root and variance contribution rate.

Element	Initial eigenvalues			Total variance explained Extract the load sum of squares			Rotational load sum of squares		
Serial	Total	Variance percent	Accumulation (%)	Total	Variance percent	Accumulation (%)	Total	Variance percent	Accumulation (%)
1	2.416	48.316	48.316	48.316	48.316	48.316	2.004	40.073	40.073
2	1.233	24.669	72.986	1.233	24.669	72.986	1.646	32.913	72.986
3	0.853	17.057	90.042	—	—	—	—	—	—
4	0.339	6.778	96.820	—	—	—	—	—	—
5	0.159	3.180	100.00	—	—	—	—	—	—

Table 3: The rotated score coefficient matrix.

Element	1	2
Total supply average	0.899	0.249
Supply stability	0.439	0.761
Effective availability	0.258	0.793
Average weekly index of actual supply	0.917	-0.009
Raw material cost	-0.310	0.613

The target function z_1 expression is as follows:

$$\min\ z_1 = \text{num}\left(\sum_{i=1}^{24} (S_A)_{ij} \neq 0\right) + \text{num}\left(\sum_{i=1}^{24} (S_B)_{ij} \neq 0\right) + \text{num}\left(\sum_{i=1}^{24} (S_C)_{ij} \neq 0\right) \tag{7}$$

(iv) Step 4: according to the above planning model, the enterprise should select at least 118 suppliers to supply raw materials to meet the demand for production, and the selected suppliers are shown in Figure 5

5.3. Model Establishment. Different from the optimization algorithm based on gradient descent commonly used in mathematical theory research [24–28], genetic algorithm is an optimization algorithm that draws on the principles of genetics and is widely used to solve transportation problems, supply chain network problems, and site selection and allocation problems [29]. Its essence is an efficient, parallel, global search method that automatically acquires and accumulates knowledge about the search space during the search process and adaptively controls the search process to find the best solution [30, 31]. The basic steps of the genetic algorithm are shown in Figure 6.

5.3.1. Coding Design. This model will use three layers of coding: (i) the selection of the least supplier in layer 1 (the 118 suppliers solved in the 5.2 model) will be coded from 0 to 1, with 0 when the supplier is not selected and 1 when the supplier is selected; (ii) the most economical ordering scheme for layer 2, which is integer coded for the supplier's supply, and each floating point number represents the supplier's supply during the week [32]; and (iii) the layer 3 loss is minimal, that is, the forwarder's choice is 0-1 coded, when a supplier chooses, 0 if the forwarder is not selected, 1 when the forwarder is selected [33].

5.3.2. Decision Variables. Let us assume that A raw material supplier, taking the first week as an example, the supply volume of each store in the first week consists of a matrix of 146 rows and 1 column as G_A, and defines G_B, G_C in the same way; at the same time, suppose that the matrix of 402 rows and 8 columns is the cooperation matrix S_Z between supplier and forwarder, which is a 0-1 matrix. Taking the first week as an example, the loss rate of each forwarder is recorded as ρ_1.

$$\rho_1 = \begin{pmatrix} 1.91 \\ 0.74 \\ \cdots \\ 0.64 \end{pmatrix}. \tag{8}$$

5.3.3. Fitness Function

(i) In order to reduce the need to reduce costs, it is planned to purchase as much class A as possible and class C raw materials as little as possible. Objective functions can be expressed by empowerment, it is advisable to give a class A raw material weight of 50, a class C raw material weight of 1, then there is a mathematical expression:

$$\max z_2 = 50 \times \sum_{i=1}^{146} (G_A) + 1 \times \sum_{i=1}^{122} (G_C) \tag{9}$$

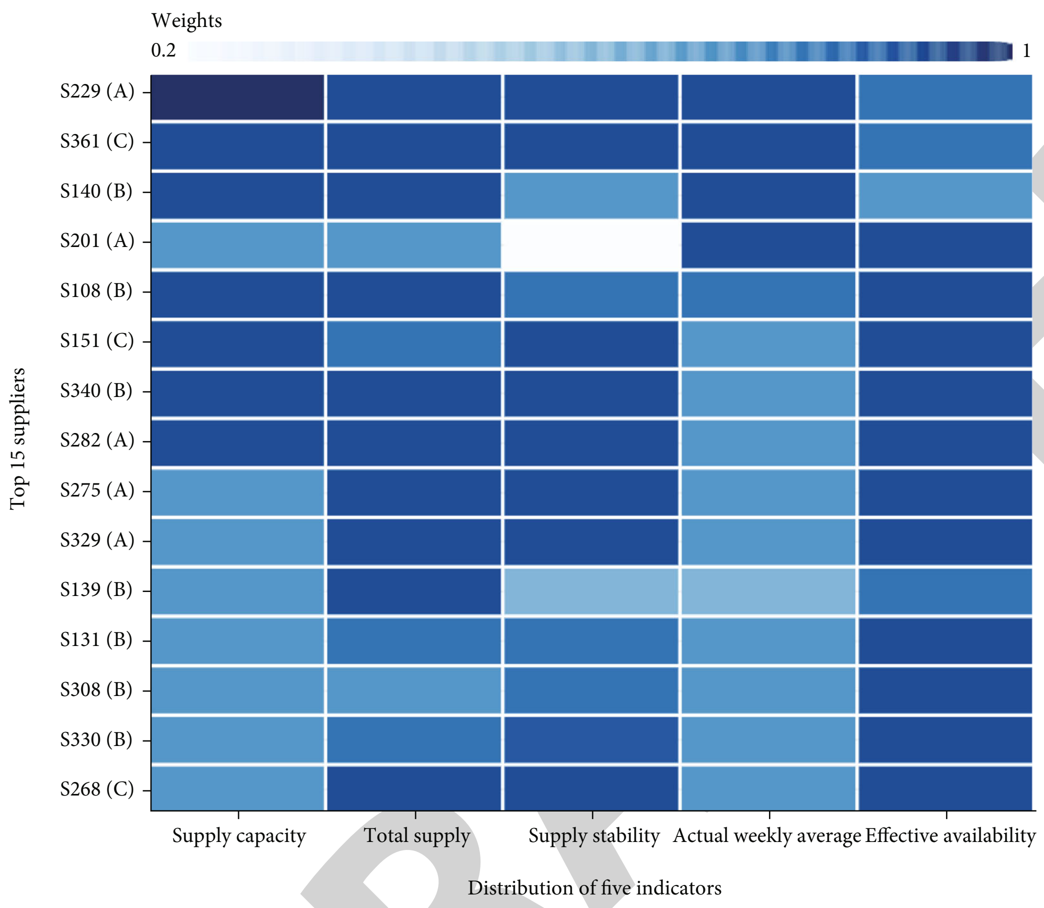


Figure 4: The top 15 most important suppliers and the proportion of each evaluation index.

(ii) To make the transshipment loss rate of the forwarder as small as possible, the expression is

$$\min z_2 = \begin{pmatrix} G_A \\ G_B \\ G_C \end{pmatrix} \cdot S_Z \cdot \rho_1 \tag{10}$$

5.3.4. Constraint Condition

(i) The most economical order plan. This article assumes that the total supply of raw materials A, B, and C is greater than the demand by 18,000 cm^3. And each week, the raw materials needed for the next week should be prepared in advance. For example, the math expression for the first week

$$95\% \times \left(\frac{\sum_{i=1}^{X} (G_A)_{ij}}{0.4} + \frac{\sum_{i=1}^{X} (G_B)_{ij}}{0.48} + \frac{\sum_{i=1}^{X} (G_C)_{ij}}{0.62} \right) \geq 3.85 \times 10^4 \tag{11}$$

The weekly supply from suppliers cannot exceed the maximum supply that their suppliers can provide.

Take the first week as an example:

$$G_A \leq \mathrm{MAX}_{A,1} \tag{12}$$

(ii) Assuming that a supplier is transported by only one forwarding company per week, and there must be a forwarding company to help it transfer:

$$\sum_{j=1}^{8} (S_Z)_{ij} = 1 \tag{13}$$

5.3.5. Generate the Initial Population. Depending on the constraints, a viable solution with a definite size is randomly generated as the initial population, with a population of 500 [34].

5.3.6. Genetic Strategies

(i) Crossover: set the crossover probability P_C of the population to 0.6. Two chromosomes are randomly selected, and when the crossover condition is met, 2 intersection points are randomly generated, and the $t1$ cycle data of chromosome 1 is exchanged with the $t2$ cycle data of chromosome 2 [35]

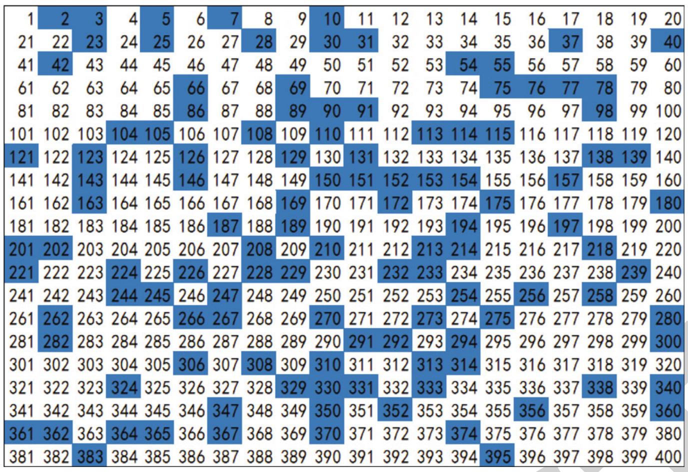


Figure 5: 118 suppliers.

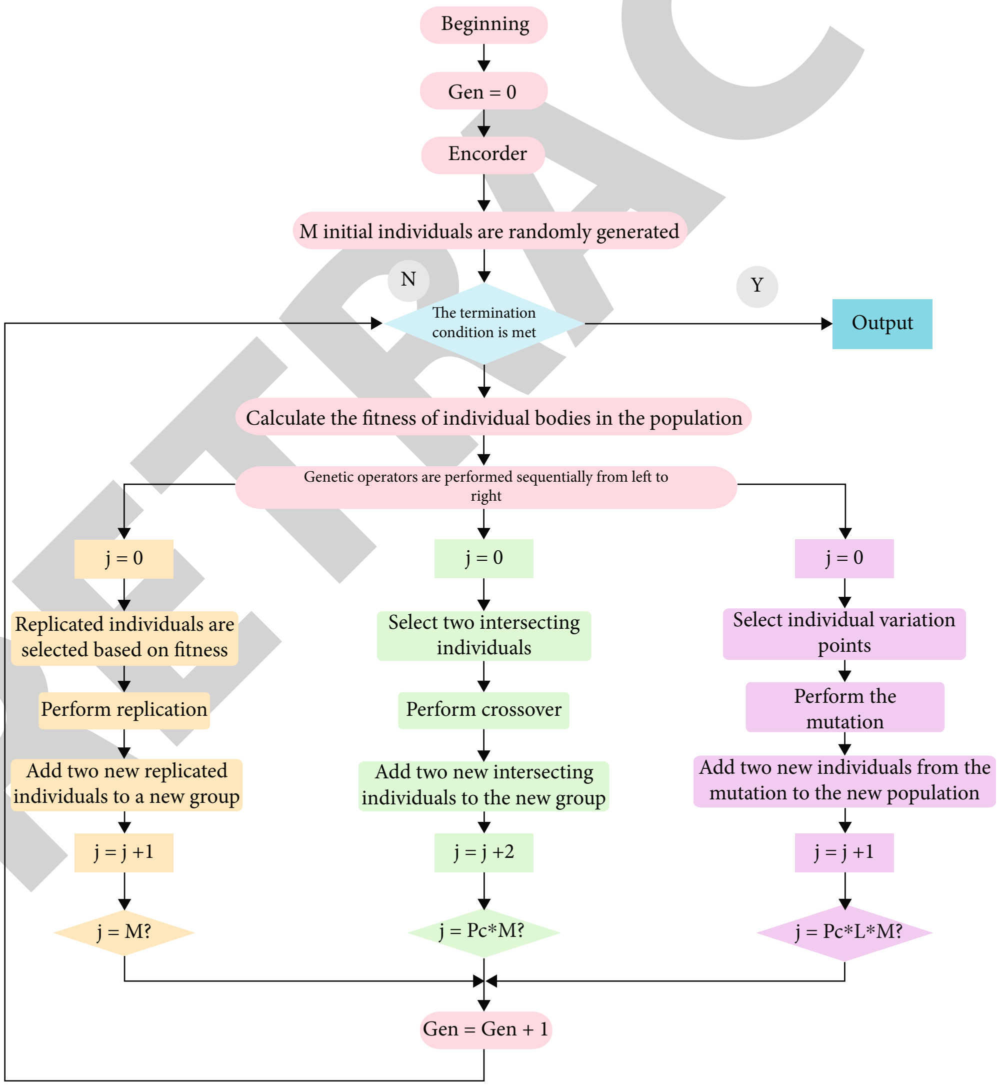

Beginning
Gen = 0
Encorder
M initial individuals are randomly generated
N
Y
The termination condition is met
Output
Calculate the fitness of individual bodies in the population
Genetic operators are performed sequentially from left to right
j = 0
Replicated individuals are selected based on fitness
Perform replication
Add two new replicated individuals to a new group
j = j +1
j = M?
j = 0
Select two intersecting individuals
Perform crossover
Add two new intersecting individuals to the new group
j = j +2
j = Pc*M?
j = 0
Select individual variation points
Perform the mutation
Add two new individuals from the mutation to the new population
j = j +1
j = Pc*L*M?
Gen = Gen + 1

Figure 6: Genetic algorithm flowchart.

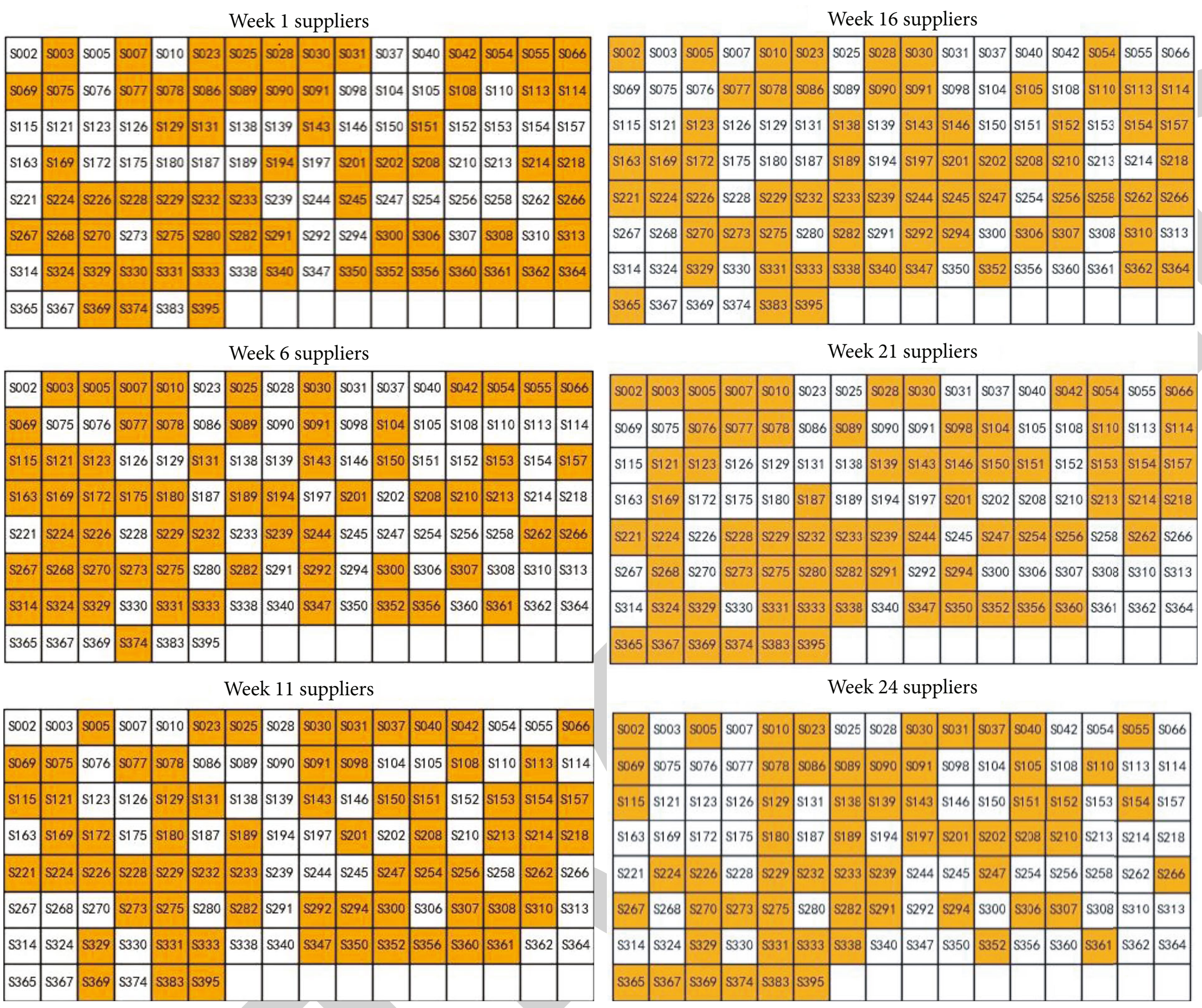


Figure 7: Supplier selection results.

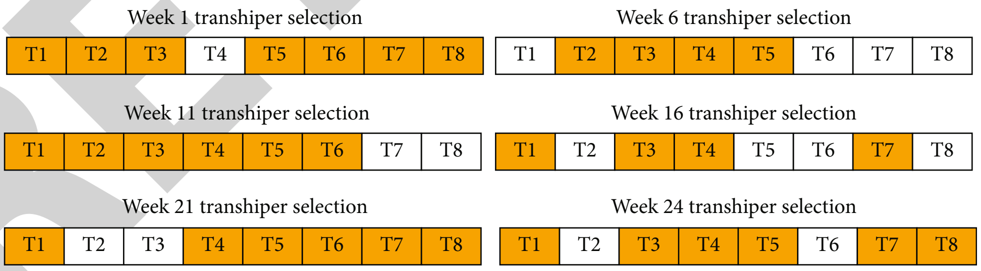


Figure 8: Transhipper selection results.

(ii) Variation: set the probability P_m of variation for the population p_m to 0. 01. A chromosome is randomly selected, and when the mutation condition is met, 2 mutation points $t1, t2 \in [1, T]$ are randomly generated, and the data of the $t1$, $t2$ cycles of the chromosome are exchanged [36, 37]

5.4. Model Solution. According to the above planning model, the company's raw material ordering and transshipment plan for the next 24 weeks and the weekly population evolution trend have been identified, due to the excessive length, here we give an example of the situation in weeks 1, 6, 11, 16, 21, and 24.

As shown in Figures 7 and 8, we can see the distribution of chromosomes, orange represents 1, and white represents 0. Under the optimization model of the genetic algorithm we designed, we provide suitable raw material suppliers and forwarders for each cycle according to the needs of enterprises, reducing the impact of subjective factors, which

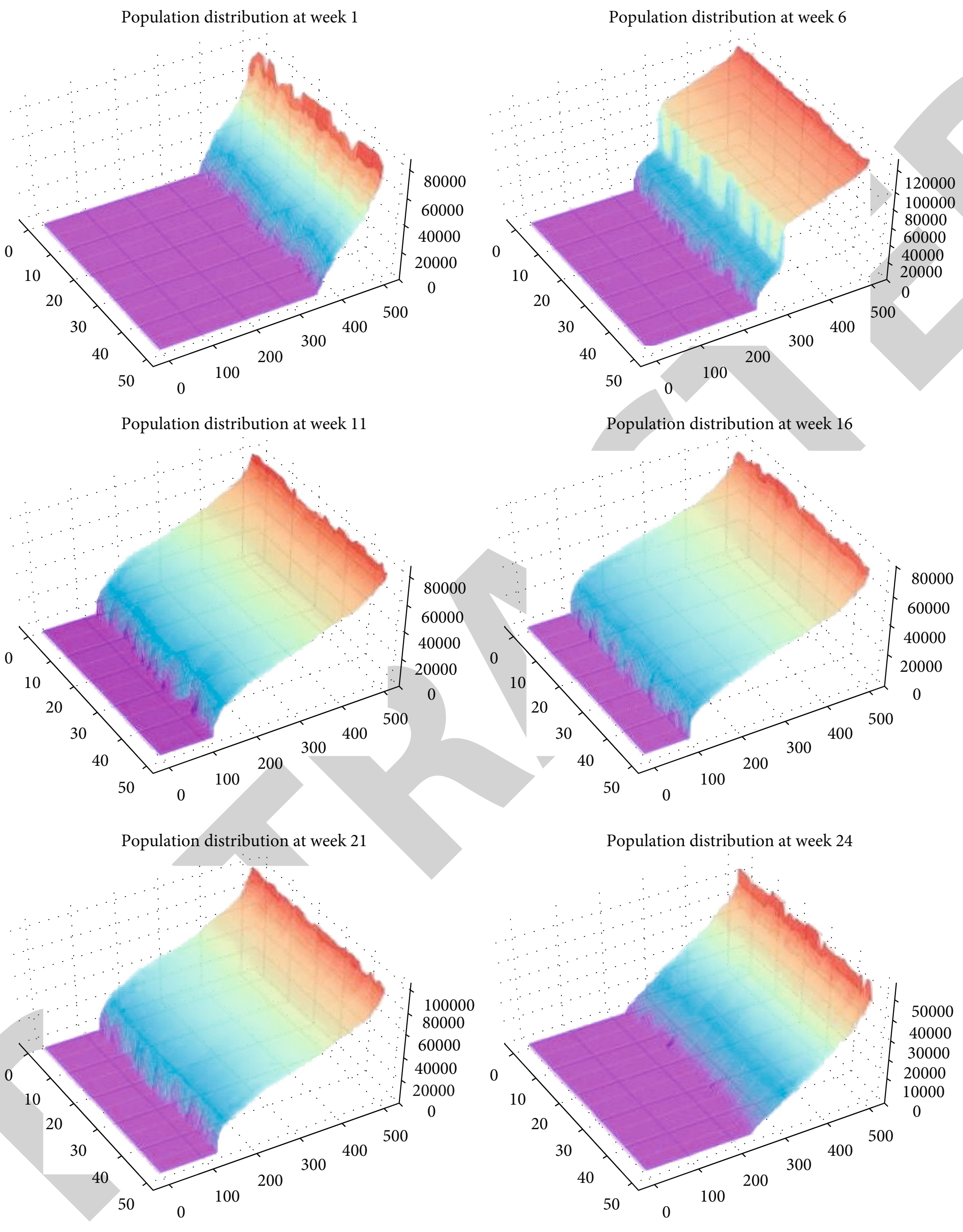


Figure 9: Population iteration results.

is conducive to enterprises to quickly lock in upstream supply chain partners, reduce costs and costs, and meet the raw material needs of enterprises in various periods [38, 39].

As shown in Figure 9, the genetic algebra used in the multiobjective genetic algorithm model we constructed is very small, with a maximum of no more than 50 generations. It can be seen that the convergence speed is very fast, which is suitable for the optimization of complex problems, and the optimal value of the supply chain system presents a cycle. It is in line with the multicycle characteristics of enterprises and has good applicability.

5.5. Calculation Verification. When verifying the model, we set the parameters of the genetic calculation as follows: the hybridization probability is 0.6, the mutation probability is 0.1, the initial population is 500, and the maximum

Table 4: Verified operation result.

Text	The algebra of iterating to the result	Initial population	Hybrid probability	Mutation probability	Best quality value
1	27	500	0.6	0.01	80921.81
2	25	500	0.6	0.01	80921.81
3	22	500	0.6	0.01	80921.81
4	26	500	0.6	0.01	80921.81
5	28	500	0.6	0.01	80921.81
6	21	500	0.6	0.01	80921.81
7	24	500	0.6	0.01	80921.81
8	25	500	0.6	0.01	80921.81

evolutionary generation is 50. For 8 consecutive calculations, the termination algebras of the 8 operations are shown in Table 4.

As shown in Table 4, the first terminates in the 22nd generation, the second terminates in the 23rd generation, the third terminates in the 24th generation, and the fourth terminates in the 25th generation. The sixth time ends at the 25th generation, the seventh time ends at the 23rd time, and the eighth time ends at the 24th time. The optimal value of the 8 calculations is 80921.81 yuan, which can prove that the genetic algorithm is effective.

6. Conclusion

In this paper, TOPSIS is used to improve the factor analysis model, and the evaluation index system to ensure the importance of enterprise production is constructed by mining 5 indicators from multiple angles, which is conducive to improving the use of raw materials by enterprises. Secondly, the genetic algorithm is used, and the constraint conditions are added, so that the objective function obtains a feasible solution under the agreed conditions, avoids local optimization, and thus achieves global optimization. By studying the optimization of the ordering and transportation of raw materials for production enterprises, it is conducive to improving the production efficiency of traditional manufacturing industries and promoting the improvement of production competitiveness of production and manufacturing industries. In view of the great advantages of deep learning technology in prediction and recommendation [40–43], in the next step, we will combine artificial intelligence technology to develop new algorithms to further optimize the ordering and transportation of raw materials.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

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

Authors' Contributions

Feng Xu was responsible for the methodology; conceptualization; supervision and leadership. Yu-Meng Zhang was responsible for the conceptualization, visualization, software, validation, and data analysis. Yi Su was responsible for writing the manuscript, verification, and investigation. Jia Li was involved in data collation verification and method design. Jia-Ming Zhu contributed to the study conception and design, supervision, and review and editing. All authors read and approved the final manuscript.

Acknowledgments

This study was funded by the National Social Science Fund Project of China (21CTJ024), and the Teaching and Research Fund Project of the Anhui University of Finance and Economics (acxkjs2021005 and acyljc2021002) and Anhui Quality Engineering Project Teaching Demonstration Course "mathematical modeling" (2020SJJXSFK0018).

References

[1] H. Chen, B. Luo, and X.-W. Yang, "An EPQ model of spoiled goods considering the inventory cost of raw materials," *Chinese Journal of Management Science*, vol. 15, no. 3, pp. 93–97, 2007.

[2] P.-J. Huo and G.-L. Xuan, "Economic ordering of raw materials and product pricing strategy," *Practice and Understanding of Mathematics*, vol. 32, no. 4, pp. 533–536, 2002.

[3] Z. Peng, S. Guo, L. Wang, J. Guo, and B. Du, "Raw materials purchasing strategy model for complex products based on time series," *IOP Conference Series: Materials Science and Engineering*, vol. 677, no. 2, p. 022108, 2019.

[4] S. Sutrisno, S. Solikhin, and P.-A. Wicaksono, "Optimisation on multi-period raw material procurement and product mixing under uncertain demand via probabilistic multi-objective model approach," *International Journal of Procurement Management*, vol. 14, no. 2, p. 147, 2021.

[5] X.-W. Wang and S. Li, "Supply chain stability evaluation of manufacturing enterprises based on projection tracing-random forest," *Operations Research and Management*, vol. 31, no. 3, pp. 171–178, 2022.

[6] Z. Dai, "Cost optimization of supply chain network and its hybrid genetic algorithm for multi-cycle and multi-raw material," *Computer Application Research*, vol. 31, no. 9, pp. 2620–2624, 2014.

[7] C.-Y. Lam, S.-L. Chan, W. H. Ip, and C. W. Lau, "Collaborative supply chain network using embedded genetic algorithms," *Industrial Management & Data Systems*, vol. 108, no. 8, pp. 1101–1110, 2008.

[8] J.-G. Huang, X.-X. Ji, and Y.-X. Li, "The impact of enterprise innovation strategy on business performance: a case study of supply chain integration strategy of manufacturing enterprises," *Scientific Management Research*, vol. 39, no. 1, pp. 111–115, 2021.

[9] L. Fang, Y. Qiu, Z. Zeng-Xiang, and G. Cai-Lan, "Study on remote sensing estimation index and model of urban ecological environment," *Journal of Infrared and Millimeter Waves*, vol. 27, no. 3, pp. 219–223, 2008.

[10] F.-G. He and H. Qi, "Nonlinear evaluation model based on principal component analysis and neural network," *Journal of Wuhan University of Technology*, vol. 29, no. 8, pp. 183–186, 2007.

[11] Y.-Y. Xu, X.-Z. Zhou, and X.-H. Jing, "Chinese sentence analysis based on maximum entropy model," *Acta Electronica Sinica*, vol. 31, no. 11, pp. 1608–1612, 2003.

[12] G.-D. Zhang, Y.-G. Xue, C.-H. Bai, M. X. Su, K. Zhang, and Y. F. Tao, "Risk assessment of floor water inrush in coal mines based on MFIM-TOPSIS variable weight model," *Journal of Central South University*, vol. 28, no. 8, pp. 2360–2374, 2021.

[13] L.-P. Yu, Q.-G. He, and Y. Han, "Pseudo-weight and weight failure of empowerment nonlinear academic evaluation methods: a case study of TOPSIS evaluation methods," *Journal of Intelligence*, vol. 41, no. 5, pp. 190–197, 2022.

[14] J.-H. Liang, Z.-B. Wang, H.-T. Zhu, and D.-Z. Sun, "Research on AHP-TOPSIS technology applicability assessment method based on water pollution control target demand," *Chinese Journal of Environmental Engineering and Technology*, vol. 12, no. 2, pp. 390–398, 2022.

[15] Z.-H. Xu and Y.-M. Cao, "Evaluation of water resource carrying capacity of Changchun City based on entropy right TOPSIS model," *Journal of Safety and Environment*, vol. 2021, p. 10, 2021.

[16] X.-L. Zhang, S.-Q. Yan, and Z.-F. Li, "Performance evaluation of farmers' professional cooperatives in underdeveloped areas: based on combination empowerment TOPSIS method," *China Journal of Agricultural Machinery and Chemistry*, vol. 43, no. 1, pp. 228–236, 2022.

[17] X.-Y. Wang, J.-F. He, F.-G. Nie, Z.-L. Yuan, and L. Lin, "X-ray fluorescence overlapping peak decomposition based on multi-adaptive metric genetic algorithm," *Spectroscopy and Spectral Analysis*, vol. 42, no. 1, pp. 152–157, 2022.

[18] S. NIKBAKHT, C. ANITESCU, and T. RABCZUK, "Optimizing the neural network hyperparameters utilizing genetic algorithm," *Journal of Zhejiang University-Science A (Applied Physics & Engineering)*, vol. 22, no. 6, pp. 407–426, 2021.

[19] W. Qi, P. Feng, B. Wei, D. Zheng, T.-T. Yu, and P.-Y. Liu, "Wavelength optimization algorithm for water quality COD detection characteristics based on embedded particle swarm-genetic algorithm," *Spectroscopy and Spectral Analysis*, vol. 41, no. 1, pp. 194–200, 2021.

[20] Z. Xu, W.-C. Ni, and Y.-H. Ji, "Rotation forest based on multimodal genetic algorithm," *Journal of Central South University*, vol. 28, no. 6, pp. 1747–1764, 2021.

[21] H.-M. Zhao, X. Zhao, F.-L. Han, and Y. L. Wang, "Cobalt crust recognition based on kernel fisher discriminant analysis and genetic algorithm in reverberation environment," *Journal of Central South University*, vol. 28, no. 1, pp. 179–193, 2021.

[22] B. Ghoulemallah, B. Sebti, C. Abdesselem, and B. Said, "Application of fuzzy PID controller based on genetic algorithm and particle swarm optimization in direct torque control of dual-star induction motor," *Journal of Central South University*, vol. 26, no. 7, pp. 1886–1896, 2019.

[23] H. F. Sadat, A. Aliakbar, and R. Bahram, "Semi-autogenous mill power prediction by a hybrid neural genetic algorithm," *Journal of Central South University*, vol. 25, no. 1, pp. 151–158, 2018.

[24] L. Lv, J. Chen, L. Zhang, and F. Zhang, "Gradient-based neural networks for solving periodic Sylvester matrix equations," *Journal of the Franklin Institute*, vol. 2022, 2022.

[25] L. Lv, J. Chen, Z. Zhang, B. Wang, and L. Zhang, "A numerical solution of a class of periodic coupled matrix equations," *Journal of the Franklin Institute*, vol. 358, no. 3, pp. 2039–2059, 2021.

[26] L. Zhang, S. Tang, and L. Lv, "An finite iterative algorithm for sloving periodic Sylvester bimatrix equations," *Journal of the Franklin Institute*, vol. 357, no. 15, pp. 10757–10772, 2020.

[27] L. Lv, Z. Zhang, L. Zhang, and X. Liu, "Gradient based approach for generalized discrete-time periodic coupled Sylvester matrix equations," *Journal of the Franklin Institute*, vol. 355, no. 15, pp. 7691–7705, 2018.

[28] L. Lv and Z. Zhang, "Finite iterative solutions to periodic Sylvester matrix equations," *Journal of the Franklin Institute*, vol. 354, no. 5, pp. 2358–2370, 2017.

[29] H.-C. Niu, D. Ji, and N.-A. Liu, "Method for optimizing the kinetic parameters for the thermal degradation of forest fuels based on a hybrid genetic algorithm," *Acta Physico-Chimica Sinica*, vol. 32, no. 9, pp. 2223–2231, 2016.

[30] J. Cheng, G.-F. Duan, Z.-Y. Liu, X. G. Li, Y. X. Feng, and X. H. Chen, "Interval multiobjective optimization of structures based on radial basis function, interval analysis, and NSGA-II," *Journal of Zhejiang University-Science A (Applied Physics & Engineering)*, vol. 15, no. 10, pp. 774–788, 2014.

[31] J.-M. Zhu, W.-Y. Xia, J.-J. Sun, J. B. Liu, and F. H. Yu, "The spread pattern on Ebola and the control schemes," *International Journal of Innovative Computing and Applications*, vol. 9, no. 2, pp. 77–89, 2018.

[32] S. M. Zhang, W. L. Zhan, H. Hu, Y. S. Liu, and J. M. Zhu, "Research on ethanol coupling to prepare C4 olefins based on BP neural network and cluster analysis," *Journal of Chemistry*, vol. 2022, Article ID 5324336, 10 pages, 2022.

[33] X.-W. Cai, Y.-Q. Bao, and M.-F. Hu, "Simulation and prediction of fungal community evolution based on RBF neural network," *Computational and Mathematical Methods in Medicine*, vol. 2021, Article ID 7918192, 13 pages, 2021.

[34] F. Xu, L. Y. Mo, H. Chen, and J. M. Zhu, "Genetic algorithm to optimize the design of high temperature protective clothing based on BP neural network," *Frontiers of Physics*, vol. 2021, article 600564, 6 pages, 2021.

[35] Q. He, P. Xia, B. Li, and J. B. Liu, "Evaluating investors' recognition abilities for risk and profit in online loan markets using nonlinear models and financial big data," *Journal of Function Spaces*, vol. 2021, Article ID 5178970, 15 pages, 2021.

[36] J.-B. Liu, T. Zhang, Y.-K. Wang, and W. Lin, "The Kirchhoff index and spanning trees of Möbius/cylinder octagonal chain," *Discrete Applied Mathematics*, vol. 307, no. 307, pp. 22–31, 2022.

[37] J. B. Liu, Y. Bao, W. T. Zheng, and S. Hayat, "Network coherence analysis on a family of nested weighted n-polygon networks," *Fractals*, vol. 29, no. 8, 2021.

[38] B. Li, H. Liang, L. Shi, and Q. He, "Complex dynamics of Kopel model with nonsymmetric response between oligopolists," *Chaos, Solitons & Fractals*, vol. 156, p. 111860, 2022.

[39] J.-M. Zhu, Y.-G. Geng, W.-B. Li, X. Li, and Q.-Z. He, "Fuzzy decision-making analysis of quantitative stock selection in VR industry based on random forest model," *Journal of Function Spaces*, vol. 2022, Article ID 7556229, 12 pages, 2022.

[40] L. Zhang, Y. Huo, Q. Ge, Y. Ma, Q. Liu, and W. Ouyang, "A privacy protection scheme for IoT big data based on time and frequency limitation," *Wireless Communications and Mobile Computing*, vol. 2021, 10 pages, 2021.

Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 4095481, 11 pages
https://doi.org/10.1155/2022/4095481

Research Article

SROBR: Semantic Representation of Obfuscation-Resilient Binary Code

Ke Tang, Zheng Shan, Fudong Liu, Yizhao Huang, Rongbo Sun, Meng Qiao, Chunyan Zhang, Jue Wang, and Hairen Gui

State Key Laboratory of Mathematical Engineering and Advanced Computing, China

Correspondence should be addressed to Zheng Shan; zzzhengming@163.com

Received 12 April 2022; Accepted 25 May 2022; Published 24 June 2022

Academic Editor: Lei Zhang

With the rapid development of information technology, the scale of software has increased exponentially. Binary code similarity detection technology plays an important role in many fields, such as detecting software plagiarism, vulnerabilities discovery, and copyright solution issues. Nevertheless, what cannot be ignored is that a variety of approaches to binary code semantic representation have been introduced recently, but few can catch up with existing code obfuscation techniques due to their maturing and extensive development. In order to solve this problem, we propose a new neural network model, named SROBR, which is a deep integration of natural language processing model and graph neural network. In SROBR, BERT is applied to capture sequence information of the binary code at the first place, and then GAT is utilized to capture the structural information. It combines natural language processing and graph neural network, which can capture the semantic information of binary programs while resisting obfuscation options in a more efficient way. Through binary code similarity detection task and obfuscated option classification task, the experimental results demonstrate that SROBR outperforms existing binary similarity detection methods in resisting obfuscation techniques.

1. Introduction

In recent years, researchers have shown an increased interest in detecting binary code similarity [1], which plays a pivotal role in program analysis. Binary code similarity detection is widely applied in many areas, including software plagiarism detection, automated vulnerability discovery, and malware identification. Existing research approaches have been intensively studied, and many remarkable achievements have been attained in this field. For example, Genius [2], Gemini [3], VulSeeker [4], InnerEye [5], Asm2Vec [6], SAFE [7], Mirrors [8], Codee [9], etc. all of these refer to the natural language processing method. They use their proposed model to embed the binary program and judge whether the binary program is similar based on the similarity of the embedded vectors.

Specifically, SAFE [7] draws on the natural language processing method, uses Word2Vec [10] to generate the embedding of instructions, then regards the assembly instructions as sequences, and uses a self-attention-based neural network to generate the embedding of binary functions. However, it did not take the structural characteristics of the binary code into explicit consideration and could represent the semantic information of the binary function preferably. Order-Matter [11] uses different models to obtain the embeddings for binary functions at three levels, semantically sensitive, structural sensitive, and order sensitive, respectively. In addition, it also defines two graph-level tasks for evaluation. DeepSemantic [12] uses BERT [13] model, which is the best model in the field of natural language processing. It mainly consists of two stages: In the pretraining stage, a general model suitable for downstream tasks is created. In the fine-tuning stage, specific models for specific tasks based on the pretrained model are generated. OSCAR [14] is based on a hierarchical transformer [15] and uses LLVM IR to capture the context information of long code sequences. InnerEYE [5] focuses on the use of neural machine translation models to solve the task of binary code similarity comparison across instruction set architectures (ISA).

However, with the development of information technology, obfuscation technology is increasingly applied to binary programs, making binary code similarity detection more and more challenging. Most research scholars have not conducted in-depth exploration on this issue. Some researchers have explored the effectiveness of the proposed method in resisting obfuscation techniques, but none of them can completely defeat it.

For instance, LoPD [16] uses a deviation-based program equivalence checking method to evaluate whether the programs are similar and have a certain degree of obfuscation-resilient ability. FOSSIL [17] recognizes malicious code functions on the basis of resisting various obfuscation techniques. In Asm2Vec [6], it shares the idea of PV-DM algorithm to detect the similarity of binary programs and manifests certain obfuscation-resilient capabilities. However, all these methods do not provide specific solutions to existing obfuscation techniques.

In order to resist obfuscation techniques better in binary code similarity detection, we find that it is a feasible scheme to combine natural language processing methods with graph processing algorithms. Inspired by DeepSemantic [12] and Order-Matters [11], we propose a new deep neural network architecture that learns the deep semantic information of binary functions by combining BERT (Bidirectional Encoder Representations from Transformers) and GAT (graph attention network). Experiments have proved that it can resist the existing binary code obfuscation technology better than the existing models.

Our contributions are concluded as follows:

(1) We adopt a new neural network architecture, named SROBR, which combines natural language processing and graph data processing. It performs well in capturing the semantic information of binary functions, while resisting the existing obfuscation methods based on the semantic information of the functions

(2) We utilize BERT model to obtain the semantic embedding of assembly instructions, so that each assembly instruction contains richer contextual information, which makes the semantic representation more accurate

(3) We apply GAT to embed the control flow graph of binary function. Through graph attention layer, we could get the attention weights and update the semantic information of the basic blocks

The remaining part of this paper proceeds as follows: Section 2 mainly introduces related work and background knowledge. Section 3 describes the structure of our proposed model in detail. Section 4 shows the results of our experiments and compares with the baselines. Section 5 summarizes the full paper.

2. Related Work

2.1. Obfuscator-LLVM Options. O-LLVM [18] is a C/C++ compiler based on the LLVM framework and Clang toolchain. It modifies the program logic at the intermediate representation level to increase the complexity of the binary while ensuring that the semantics remain unchanged. This feature can not only protect the copyright of the software and prevent it from being disassembled and analyzed by others but also hide the actual purport of the software itself, thereby to carry out malicious behaviors.

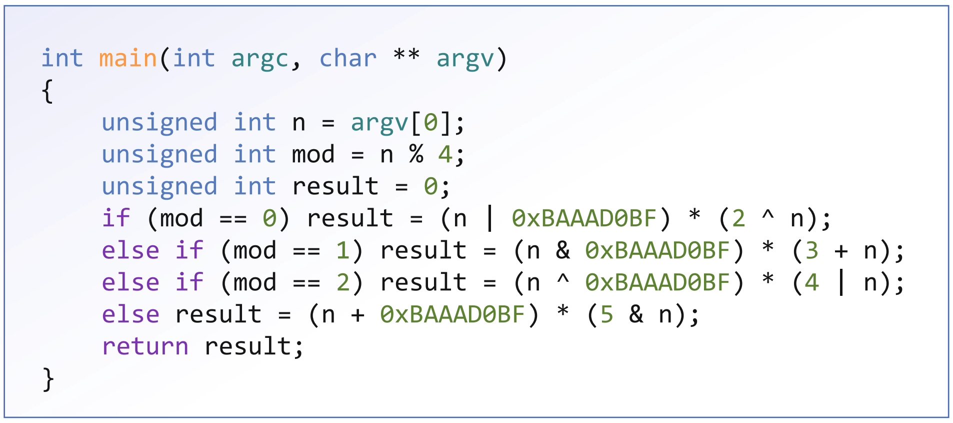
There are three obfuscation options for application at present. The *SUB* (Instruction Substitution) option will replace part of the assembly instructions with equivalent code fragments without changing the structural information of control flow graph. The *BCF* (bogus control flow) option will generate bogus control flow by adding invalid edges and nodes of the function control flow graph. The *FLA* (Control Flow Flattening) option will use a complex hierarchical structure to reconstruct the control flow graph and fuse it into a linear structure while ensuring that its semantics remain unchanged.

We use a simple example to intuitively exemplify the effect of these three obfuscation options. The source code is shown in Figure 1. Figure 2 shows the different program control flow graph (CFG) by compiling the same source code with different obfuscation options. Figure 1 is the CFG compiled without any obfuscation options. Figure 1 uses the *sub* option, and its CFG structure has basically not unchanged. Only some instructions are replaced with more complex equivalent instructions. Figure 1 uses the *bcf* option, of which the CFG is very different from (a), and introduces a lot of false instruction blocks. Figure 1 uses the *fla* option to completely disrupt the basic blocks in the CFG. It is difficult to understand the semantics of the program by using traditional reverse analysis.

2.2. Existing Approaches. LoPD [16] proposes a deviation-based program equivalence checking method, which searches for any dissimilarity between two programs by finding an input that will lead these two programs to be having differently, either with different output states or with semantically different execution paths. However, this method will consume a lot of time and cannot be applied on a large scale, and the results may be distorted.

FOSSIL [17] integrates a range of syntactical, semantic, and behavioral features by using Bayesian network model, which can recognize the functions in the open source software for the malicious code. Besides, it has the ability to resist obfuscation options and compiler optimization options. But it did not focus on the similarity of obfuscated code.

Asm2Vec [6] proposes to model the control flow graph as multiple sequences by using random walk algorithm. Each sequence corresponds to a potential execution trace that contains linearly laid-out assembly instructions. Then, these sequences are taken as input, and the PV-DM [19] model are used for training to learn the semantic representation of the function. Converting CFG to sequences will lose the structural information of the program, which causes this model not capable of representing the semantic information



```c
int main(int argc, char ** argv)
{
    unsigned int n = argv[0];
    unsigned int mod = n % 4;
    unsigned int result = 0;
    if (mod == 0) result = (n | 0xBAAAD0BF) * (2 ^ n);
    else if (mod == 1) result = (n & 0xBAAAD0BF) * (3 + n);
    else if (mod == 2) result = (n ^ 0xBAAAD0BF) * (4 | n);
    else result = (n + 0xBAAAD0BF) * (5 & n);
    return result;
}
```

Figure 1: The source code of test function.

of the function completely. Although this method shows certain resistance to obfuscation options, it cannot completely defeat the code obfuscation.

2.3. BERT. BERT [13] is currently the best performing pretraining model in the field of natural language processing. It is different from the traditional monodirectional language model or the shallow splicing of two monodirectional language models. Instead, its main architecture is a stack of transformer's encoders [15], each of these layers utilizes the self-attention mechanism to learn the semantic representation of natural language. BERT is actually a two-stage framework, including pretraining and fine-tuning. First, it performs pretraining on a large corpus to obtain a generalized representation and then fine-tune it for specific tasks. A large number of experiments have proved that this method can achieve good results and the same in the field of assembly language analysis. For example, Order-Matter [11] and DeepSemantic [12] have used the BERT model and achieved good results. Therefore, we will also use the BERT model to learn deeper semantic information in the binary function to resist the obfuscation options of O-LLVM.

2.4. GAT. Although traditional deep learning algorithms have been applied to extract the features of Euclidean spatial data with great success, its performance on dealing with the data that generated from non-Euclidean spaces in many practical scenarios is not satisfactory. This is because the graph is irregular, each graph has unordered nodes of variable size, and each node in the graph has a different number of adjacent nodes, which makes it difficult for existing deep learning algorithms to deal with it. In addition, a core assumption of existing deep learning algorithms is that the data samples are independent of each other. However, each data sample (node) has edges related to other real data samples (nodes) in the graph, and this point can be used to scout the interdependence relations between nodes.

In recent years, people have become more and more interested in the expansion of deep learning algorithms on graphs. Successfully driven by many factors, the researchers integrated the ideas of convolutional networks, recurrent networks, and deep autoencoders to define and design the neural network structure for processing graph data, which brought up the graph neural network.

Graph neural networks mainly include graph convolution networks (GCN) [20, 21], graph attention networks (GAT) [22], graph autoencoders (GAE), [23] etc. In the field of semantic representation of binary codes, Qiao et al. [24] and Massarelli et al. [25] use GCN for semantic embedding of functions, but considering that the CFG of functions is a directed graph, which will cause a certain loss of the structural information. To avoid this problem, we adopt the GAT model instead.

GAT is a spatial-based graph convolutional network. It uses the attention mechanism to determine the weights of neighbor nodes when aggregating feature information. The GAT [22] introduces a self-attention mechanism in the propagation process, and the hidden state of each node is calculated in consideration of its neighbor nodes. In the internal structure of the GAT network [22], it is a simple stack of graph attention layers. For the node pair (i, j) in each attention layer, the attention coefficient is calculated as

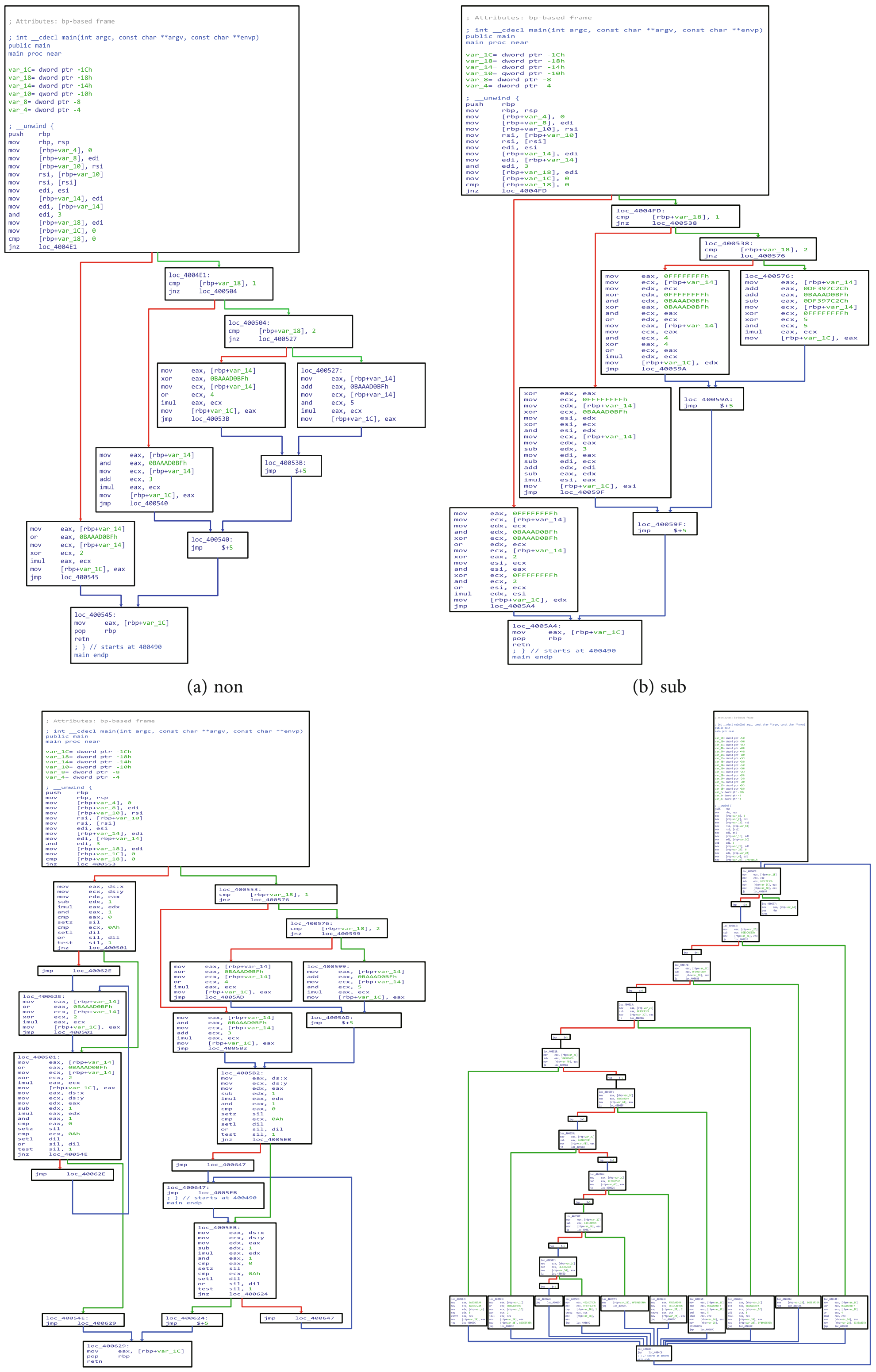
$$\alpha_{ij} = \frac{\exp\left(LeakyReLU\left(a^T\left[Wh_i \| Wh_j\right]\right)\right)}{\sum_{k\in N_i} \exp\left(LeakyReLU\left(a^T\left[Wh_i \| Wh_k\right]\right)\right)}, \quad (1)$$

where N_i represents the neighbors of node i, the input feature of the nodes is $h = \{h_1, h_2, h_3, \cdots, h_n\}$, $h_i \in \mathbb{R}^F$, and N, F represent the number of nodes and the feature dimension, respectively. The output feature of the nodes is $h' = \{h'_1, h'_2, h'_3, \cdots, h'_n\}$, $h'_i \in \mathbb{R}^{F'}$, in which F' is the output feature dimension. $W \in R^{F' \times F}$ is the linear transformation weight matrix on each node, and $a \in R^{2F'}$ is the weight vector. Finally, the Softmax function is used for normalization, and Leaky*ReLU* is added to provide nonlinearity.

The finally output feature of the node i is

$$h'_i = \sigma\left(\sum_{j\in N_i} \alpha_{ij} W h_j\right). \quad (2)$$

Multihead attention can be applied in GAT [22] to enhance the learning ability of the model. It applies k



```
; Attributes: bp-based frame

; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near

var_1C= dword ptr -1Ch
var_18= dword ptr -18h
var_14= dword ptr -14h
var_10= qword ptr -10h
var_8= dword ptr -8
var_4= dword ptr -4

; __unwind {
push    rbp
mov     rbp, rsp
mov     [rbp+var_4], 0
mov     [rbp+var_8], edi
mov     [rbp+var_10], rsi
mov     rsi, [rbp+var_10]
mov     rsi, [rsi]
mov     edi, esi
mov     [rbp+var_14], edi
mov     edi, [rbp+var_14]
and     edi, 3
mov     [rbp+var_18], edi
mov     [rbp+var_1C], 0
cmp     [rbp+var_18], 0
jnz     loc_4004E1

loc_4004E1:
cmp     [rbp+var_18], 1
jnz     loc_400504

loc_400504:
cmp     [rbp+var_18], 2
jnz     loc_400527

mov     eax, [rbp+var_14]
xor     eax, 0BAAAD0BFh
mov     ecx, [rbp+var_14]
or      ecx, 4
imul    eax, ecx
mov     [rbp+var_1C], eax
jmp     loc_40053B

loc_400527:
mov     eax, [rbp+var_14]
add     eax, 0BAAAD0BFh
mov     ecx, [rbp+var_14]
and     ecx, 5
imul    eax, ecx
mov     [rbp+var_1C], eax

mov     eax, [rbp+var_14]
and     eax, 0BAAAD0BFh
mov     ecx, [rbp+var_14]
add     ecx, 3
imul    eax, ecx
mov     [rbp+var_1C], eax
jmp     loc_400540

loc_40053B:
jmp     $+5

mov     eax, [rbp+var_14]
or      eax, 0BAAAD0BFh
mov     ecx, [rbp+var_14]
xor     ecx, 2
imul    eax, ecx
mov     [rbp+var_1C], eax
jmp     loc_400545

loc_400540:
jmp     $+5

loc_400545:
mov     eax, [rbp+var_1C]
pop     rbp
retn
; } // starts at 400490
main endp
```

(a) non

```
; Attributes: bp-based frame

; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near

var_1C= dword ptr -1Ch
var_18= dword ptr -18h
var_14= dword ptr -14h
var_10= qword ptr -10h
var_8= dword ptr -8
var_4= dword ptr -4

; __unwind {
push    rbp
mov     rbp, rsp
mov     [rbp+var_4], 0
mov     [rbp+var_8], edi
mov     [rbp+var_10], rsi
mov     rsi, [rbp+var_10]
mov     rsi, [rsi]
mov     edi, esi
mov     [rbp+var_14], edi
mov     edi, [rbp+var_14]
and     edi, 3
mov     [rbp+var_18], edi
mov     [rbp+var_1C], 0
cmp     [rbp+var_18], 0
jnz     loc_4004FD

loc_4004FD:
cmp     [rbp+var_18], 1
jnz     loc_400538

loc_400538:
cmp     [rbp+var_18], 2
jnz     loc_400576

mov     eax, 0FFFFFFFFh
mov     ecx, [rbp+var_14]
mov     edx, ecx
xor     edx, 0FFFFFFFFh
and     edx, 0BAAAD0BFh
xor     eax, 0BAAAD0BFh
and     ecx, eax
or      edx, ecx
mov     eax, [rbp+var_14]
mov     ecx, eax
and     ecx, 4
xor     eax, 4
or      ecx, eax
imul    edx, ecx
mov     [rbp+var_1C], edx
jmp     loc_40059A

loc_400576:
mov     eax, [rbp+var_14]
add     eax, 0DF397C2Ch
add     eax, 0BAAAD0BFh
sub     eax, 0DF397C2Ch
mov     ecx, [rbp+var_14]
xor     ecx, 0FFFFFFFFh
xor     ecx, 5
and     ecx, 5
imul    eax, ecx
mov     [rbp+var_1C], eax

xor     eax, eax
mov     ecx, 0FFFFFFFFh
mov     edx, [rbp+var_14]
xor     ecx, 0BAAAD0BFh
mov     esi, edx
xor     esi, ecx
and     esi, edx
mov     ecx, [rbp+var_14]
mov     edx, eax
sub     edx, 3
mov     edi, eax
sub     edi, ecx
add     edx, edi
sub     eax, edx
imul    esi, eax
mov     [rbp+var_1C], esi
jmp     loc_40059F

loc_40059A:
jmp     $+5

mov     eax, 0FFFFFFFFh
mov     ecx, [rbp+var_14]
mov     edx, ecx
and     edx, 0BAAAD0BFh
xor     ecx, 0BAAAD0BFh
or      edx, ecx
mov     ecx, [rbp+var_14]
xor     eax, 2
mov     esi, ecx
and     esi, eax
xor     ecx, 0FFFFFFFFh
and     ecx, 2
or      esi, ecx
imul    edx, esi
mov     [rbp+var_1C], edx
jmp     loc_4005A4

loc_40059F:
jmp     $+5

loc_4005A4:
mov     eax, [rbp+var_1C]
pop     rbp
retn
; } // starts at 400490
main endp
```

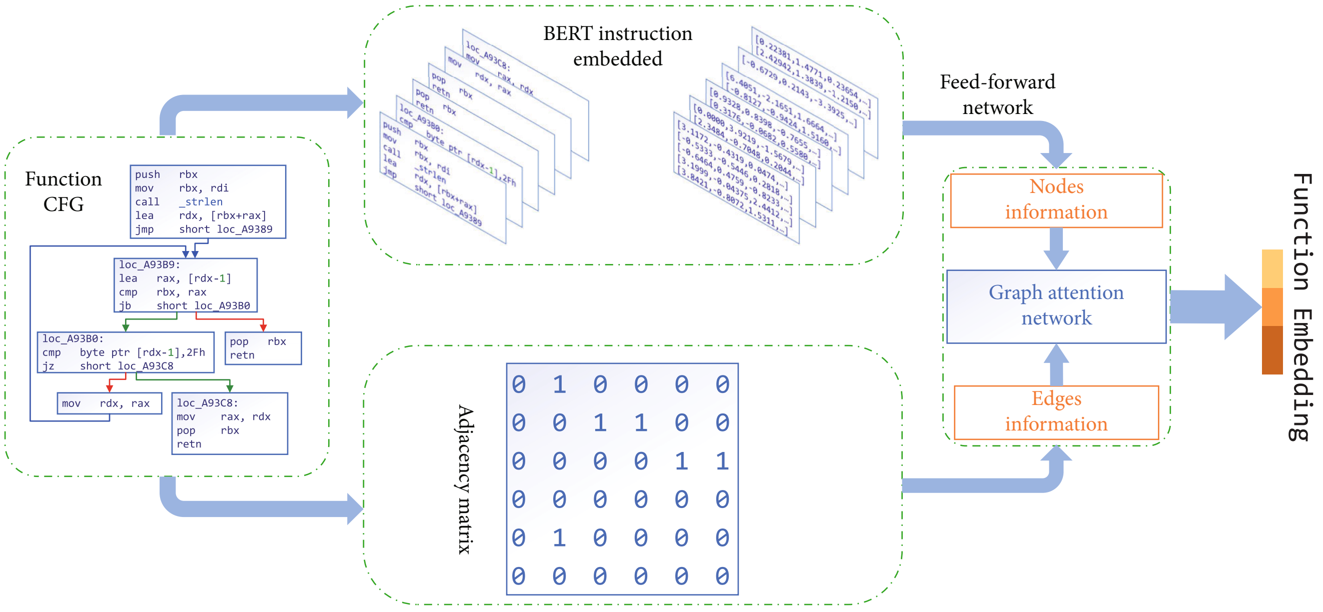
(b) sub

```
; Attributes: bp-based frame

; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near

var_1C= dword ptr -1Ch
var_18= dword ptr -18h
var_14= dword ptr -14h
var_10= qword ptr -10h
var_8= dword ptr -8
var_4= dword ptr -4

; __unwind {
push    rbp
mov     rbp, rsp
mov     [rbp+var_4], 0
mov     [rbp+var_8], edi
mov     [rbp+var_10], rsi
mov     rsi, [rbp+var_10]
mov     rsi, [rsi]
mov     edi, esi
mov     [rbp+var_14], edi
mov     edi, [rbp+var_14]
and     edi, 3
mov     [rbp+var_18], edi
mov     [rbp+var_1C], 0
cmp     [rbp+var_18], 0
jnz     loc_400553

mov     eax, ds:x
mov     ecx, ds:y
mov     edx, eax
sub     edx, 1
imul    eax, edx
and     eax, 1
cmp     eax, 0
setz    sil
cmp     ecx, 0Ah
setl    dil
or      sil, dil
test    sil, 1
jnz     loc_400501

loc_400553:
cmp     [rbp+var_18], 1
jnz     loc_400576

jmp     loc_40062E

loc_400576:
cmp     [rbp+var_18], 2
jnz     loc_400599

mov     eax, [rbp+var_14]
xor     eax, 0BAAAD0BFh
mov     ecx, [rbp+var_14]
or      ecx, 4
imul    eax, ecx
mov     [rbp+var_1C], eax
jmp     loc_4005AD

loc_400599:
mov     eax, [rbp+var_14]
add     eax, 0BAAAD0BFh
mov     ecx, [rbp+var_14]
and     ecx, 5
imul    eax, ecx
mov     [rbp+var_1C], eax

loc_40062E:
mov     eax, [rbp+var_14]
or      eax, 0BAAAD0BFh
mov     ecx, [rbp+var_14]
xor     ecx, 2
imul    eax, ecx
mov     [rbp+var_1C], eax
jmp     loc_400501

mov     eax, [rbp+var_14]
and     eax, 0BAAAD0BFh
mov     ecx, [rbp+var_14]
add     ecx, 3
imul    eax, ecx
mov     [rbp+var_1C], eax
jmp     loc_4005B2

loc_4005AD:
jmp     $+5

loc_400501:
mov     eax, [rbp+var_14]
or      eax, 0BAAAD0BFh
mov     ecx, [rbp+var_14]
xor     ecx, 2
imul    eax, ecx
mov     [rbp+var_1C], eax
mov     eax, ds:x
mov     ecx, ds:y
mov     edx, eax
sub     edx, 1
imul    eax, edx
and     eax, 1
cmp     eax, 0
setz    sil
cmp     ecx, 0Ah
setl    dil
or      sil, dil
test    sil, 1
jnz     loc_40054E

loc_4005B2:
mov     eax, ds:x
mov     ecx, ds:y
mov     edx, eax
sub     edx, 1
imul    eax, edx
and     eax, 1
cmp     eax, 0
setz    sil
cmp     ecx, 0Ah
setl    dil
or      sil, dil
test    sil, 1
jnz     loc_4005EB

jmp     loc_400647

jmp     loc_40062E

loc_400647:
jmp     loc_4005EB
; } // starts at 400490
main endp

loc_4005EB:
mov     eax, ds:x
mov     ecx, ds:y
mov     edx, eax
sub     edx, 1
imul    eax, edx
and     eax, 1
cmp     eax, 0
setz    sil
cmp     ecx, 0Ah
setl    dil
or      sil, dil
test    sil, 1
jnz     loc_400624

loc_40054E:
jmp     loc_400629

loc_400624:
jmp     $+5

jmp     loc_400647

loc_400629:
mov     eax, [rbp+var_1C]
pop     rbp
retn
```

(c) bcf

(d) fla

Figure 2: CFGs compiled with different obfuscation options.



```
push   rbx
mov    rbx, rdi
call   _strlen
lea    rdx, [rbx+rax]
jmp    short loc_A9389

loc_A93B9:
lea   rax, [rdx-1]
cmp   rbx, rax
jb    short loc_A93B0

loc_A93B0:
cmp   byte ptr [rdx-1],2Fh
jz    short loc_A93C8

pop   rbx
retn

mov   rdx, rax

loc_A93C8:
mov    rax, rdx
pop    rbx
retn
```

```
0 1 0 0 0 0
0 0 1 1 0 0
0 0 0 0 1 1
0 0 0 0 0 0
0 1 0 0 0 0
0 0 0 0 0 0
```

Figure 3: The overall architecture of SROBR.

independent attention mechanisms to calculate the hidden state and then stitches or averages the features, such as

$$h_i' = \|_{k=1}^{K} \sigma\left(\sum_{j\in N_i} \alpha_{ij}^k W^k h_j\right), \tag{3}$$

or

$$h_i' = \sigma\left(\frac{1}{k}\sum_{k=1}^{K} \sum_{j\in N_i} \alpha_{ij}^k W^k h_j\right), \tag{4}$$

where α_{ij}^k is the attention coefficient of the kth attention head.

3. Model Design

3.1. Overview. The overall framework of SROBR is shown in Figure 3. The model architecture SROBR is mainly composed of three parts: The first part draws on the method of natural language processing, takes basic blocks as input, and uses the BERT model to obtain the instruction embedding according to the context information of the instruction. The second part uses feed-forward network to aggregate the embeddings of the instructions in the basic block and uses nonlinear mapping as the semantic representation of the basic block. The third part uses the graph attention neural network, takes the embeddings of the basic block as vertices, and uses the adjacency matrix of the control flow graph as the edges to obtain the high-dimensional vector containing the semantic information of the entire function. SROBR can be formally described as

$$\text{embdding}_f = \text{LayerNorm}\left(\text{GAT}\left(\text{FFN}\left(\sum_{i\in b}\text{BERT}(i)\right), \text{adj}_f\right)\right), \tag{5}$$

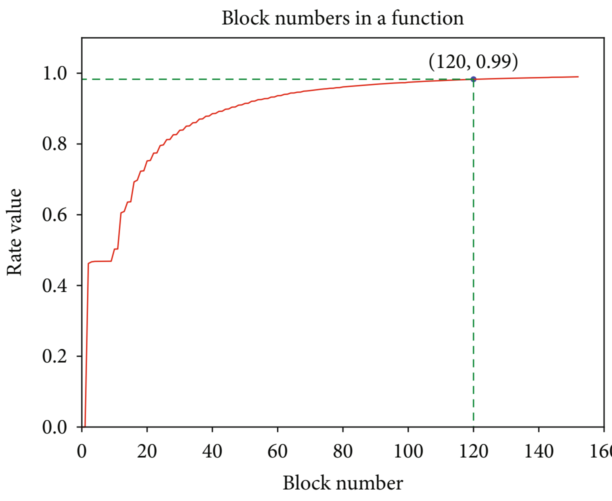


Figure 4: Basic block numbers.

where $i \in b$ represents all the instructions i in the basic block b, and adj_f represents the adjacency matrix of the basic block in the function f.

In general, SROBR takes the CFG of the binary function as input and obtains a high-dimensional vector to represent its semantic information. This digitized vector is proved to be resistant to obfuscation in subsequent experiments.

3.2. Data Preprocess. Through statistics on the basic information of our dataset, from Figure 4, we find that the number of basic blocks does not exceed 120 in more than 99% of functions. Since SROBR uses a graph neural network, too many basic blocks will lead to a sharp increase in memory usage. Limited by our hardware environment, we discard functions with more than 120 basic blocks. In addition, from Figure 5, we find that more than 99% of the basic blocks have no more than 40 instructions, so we define the maximum number of

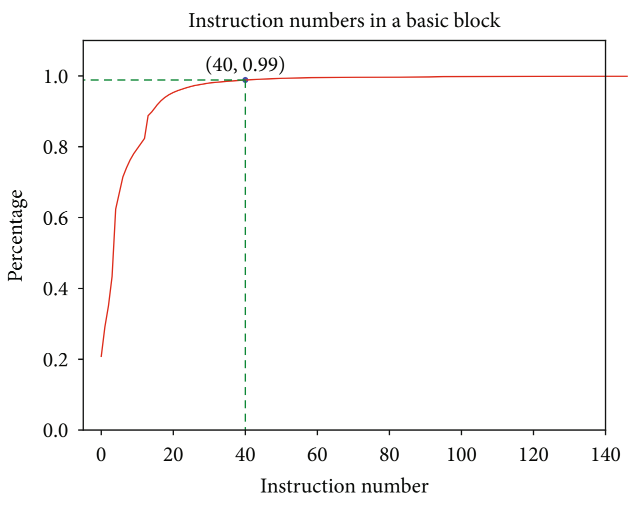


Figure 5: Instruction numbers.

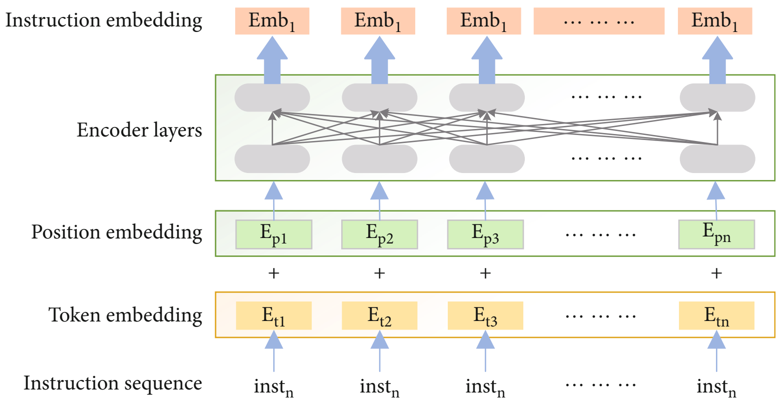


Figure 6: The structure of the instruction embedded module.

instructions in the basic block as 40. If the number of instructions in the basic blocks is more than 40, it will be truncated. Through processing, the amount of parameters can be significantly reduced, so that we can learn as much semantic information as possible while training.

3.3. Instruction Standardization. We learn from the natural language processing method to process the assemblers, in order to avoid OOV problems, we need to standardize the assembly instructions. In the standardization process, we first preprocess the assembly files to remove all comments, invalid characters, useless strings, and other unwanted content. For every instruction, it may contain various registers, memory addresses, variable names, immediate numbers, and other auxiliary information. We need to make a trade-off between the vocabulary size and the information retained.

Inspired by the normalization process in DeepSemantic [12], we propose the following regularization rules: we standardize the registers according to their categories and numbers of bits. For example, we replace "rax" with "reg_data_64," replace "edi" with "reg_addr_32," and replace "rbp" with "reg_pointer." In addition, the names of the variables in the instruction do not convey too much semantic information,

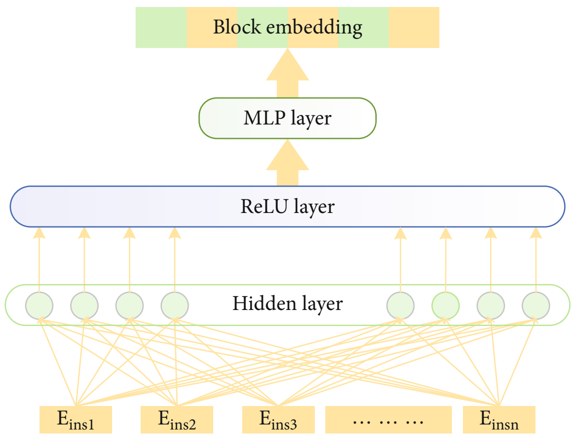


Figure 7: The structure of the feed-forward layer.

so we use "var" to replace them. For immediate numbers, we use "imm" to replace them with. Through our instruction standardization process, although small amount of the information will be lost, it can significantly reduce the noise data,

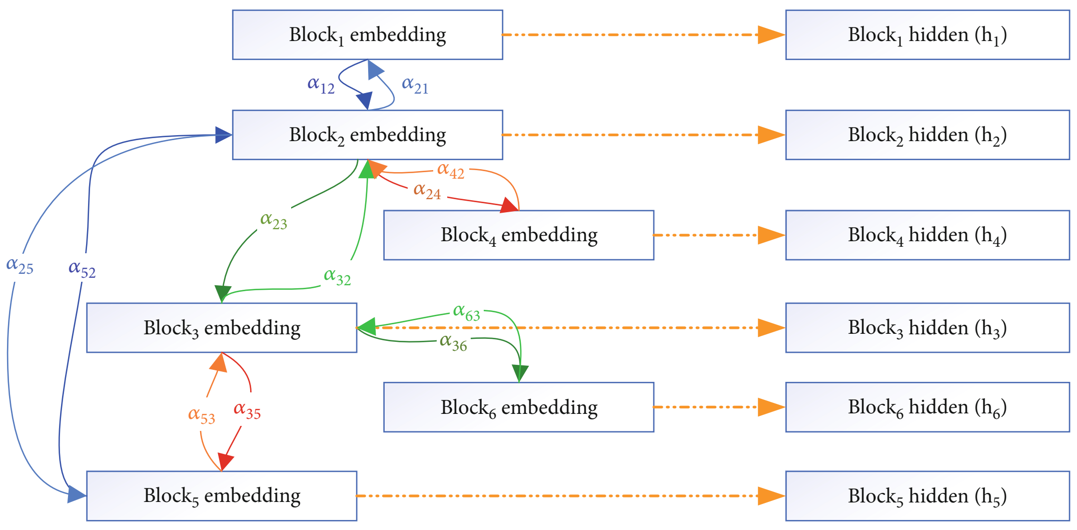


Figure 8: Graph attention weights propagation process.

which will make SROBR learn the semantic information of the function preferably.

3.4. Instruction Semantic Embedding. In the first step, we use BERT to embed the instructions in the basic block to obtain semantic information. In most papers that study the semantic representation of assembly language, such as "Asm2Vec," they generate assembly instruction embedding through word2vec. In this way, regardless of the CBOW or skip-gram method, the generated embedding is fixed and will not be affected by its contextual information. But in a basic block, the execution of the current instruction may be affected by the execution result of the previous instruction and may also affect the execution of the subsequent instruction. Therefore, we use the BERT pretraining model to embed the current instruction according to the context information of the instruction.

The structure of the assembly instruction embedded module is shown in Figure 6. In this module, each instruction in the basic block is taken as input, token embedding and positional embedding are added to educe the initial instruction vector, and the final output is attained through several encoder layers.

3.5. Block Semantic Embedding. Through the BERT model in the previous section, we get the semantic embedding of the instruction, and then we will get the semantic embedding of the basic block. Applying the feed-forward Neural network, with all the instruction embeddings in a basic block as input, we use the fully connected layer and the nonlinear mapping to semantically aggregate the instruction embeddings, so as to get the semantic information of the basic block. The basic principle is shown in Figure 7.

In this module, we first use the fully connected layer to map the embeddings to a higher-dimensional vector space and then use the RELU activation function as a nonlinear layer. Then, we use the fully connected layer to map to the original vector space and add the vectors as the embedding of the basic block.

3.6. Function Semantic Embedding. After getting the semantic embedding of the basic blocks, we are ready to obtain the basic block embedding at the function level. In this section, we will use graph attention neural network for training to obtain a vector representation containing the semantics of the entire binary function.

Graph attention neural network (GAT) introduces a self-attention mechanism in the propagation process, and the hidden state of each node is calculated by paying attention to its neighbor nodes. The control flow graph of a binary function is a directed graph, in which each node is a basic block. These nodes are linked by jump instructions. The execution of instructions in each basic block may be affected by neighbor blocks.

Therefore, we propose to use the attention mechanism to simulate the effect between blocks through attention weights, so as to restore the semantic information of the function better, even when the obfuscation technology is applied. Figure 8 visually shows the working principle of graph attention weights.

4. Experimental Evaluation

4.1. Dataset Collection. Like Asm2Vec [6], we also use commonly open source projects on github as our dataset, including 8 projects such as OpenSSL (https://github.com/openssl/openssl), libGmp (https://github.com/libtom/libtomcrypt), libTomCrypt (https://github.com/mirror/busybox), SQLite (https://github.com/curl/curl), Busybox (https://rada.re/n/radare2.html), Diffutils (http://angr.io/), Libcurl (https://rada.re/n/radare2.html), and Zlib (http://angr.io/). Specific details about the dataset are shown in Table 1.

There are two ways to generate assembler files from source code. One is to compile the source code into binary files and then use a disassembly tool (such as Radare2 (https://rada.re/n/radare2.html) or Angr (http://angr.io/)) to analyze the binaries to get the assembler files. The another one is to directly compile the source code into assemblers. The assembler files generated by the two methods are almost the same, but the latter is more convenient, so we choose the second method.

Table 1: Detailed description of our dataset.

	Function numbers	Block numbers	Instruction numbers
OpenSSL	11384	221564	1477384
LibGmp	8760	187600	1087912
LibTomCrypt	2088	54972	436844
SQLite	1464	29732	174616
Busybox	200	4264	21108
Diffutils	632	12252	63820
LibCurl	180	3936	23968
Zlib	728	16488	97784
Total	25436	530808	3383436

Then, we use the four obfuscation options of O-LLVM to compile, including non, sub, fla, bcf (non means that the obfuscation option is not used, and the source code is compiled normally). In this way, we can get the initial dataset, in which there are four binary functions with different obfuscation options for each source code.

When we conduct an in-depth research on the obtained dataset, we find that because some functions are too simple, regardless of whether it is obfuscated or not, the corresponding assemblies are exactly the same. Given that these functions may interfere with our subsequent model training, we just filter them. At last, we get more than 11000 functions for each obfuscation option.

4.2. Model Training. In this section, we apply our dataset to train the model. In SROBR, we use triple loss and stochastic gradient descent to pretrain.

In order to explore the effect of hyperparameters on the training effect of the model, we train the proposed model with different dimensions and compare the training results, as shown in Figures 9 and 10. From Figures 9 and 10, we can see that the training effect of the model gradually gets better with the increase of the dimension, but when the dimension reaches 768, the effect of the test loss is worse than that of the dimension 512. Therefore, we choose 512 as the dimension of the model.

In addition, in BERT module, we use 12 encoder layers, 8 attention heads, and dropout set 0.1. In GAT module, we use 8 graph attention layers, with dropout set 0.1 and alpha set 0.2, then we perform *LayerNorm* on the output. The main hyperparameters we set are shown in Table 2.

4.3. Evaluation. We use two tasks to evaluate our model. The first one is the binary similarity comparison task. In a round of comparison, we randomly sample 100 functions from all datasets as the current test set and randomly select a function from the current test set as the objective function. Then, we compile each function in the current test set without the obfuscation option to obtain the corresponding unobfuscated assembly functions as our set of search functions. At the same time, we compile the objective function with the specific obfuscation option to obtain the corresponding assembly function, as the target function. We use our pretrained model

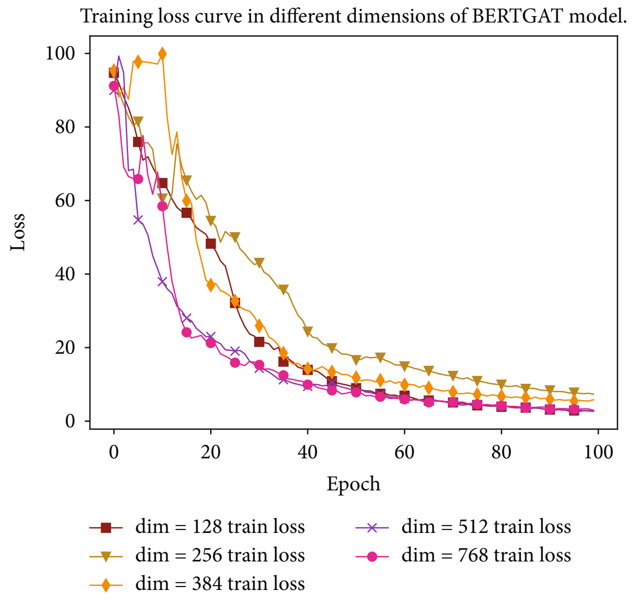


Figure 9: Training loss in different dimensions of SROBR model.

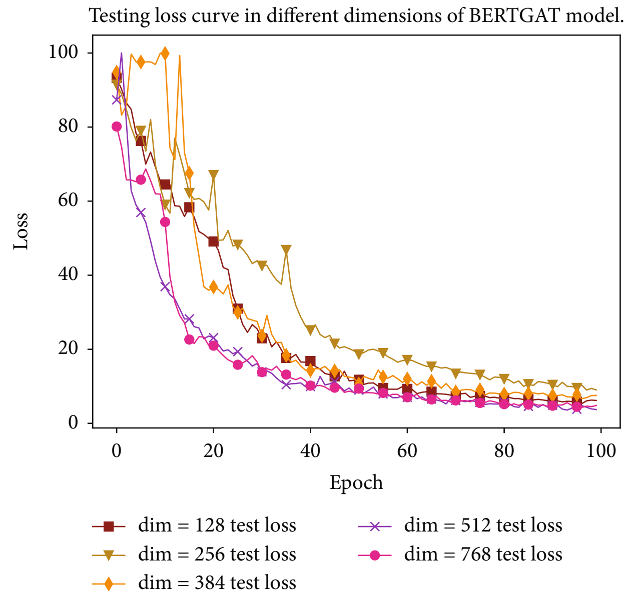


Figure 10: Testing loss in different dimensions of SROBR model.

to obtain the semantic embedding vectors of these functions, and compare the similarity between the target function and each function in the search function set according to the vector, and sort according to the similarity. Here, we use Euclidean distance as the similarity criterion. We assume that the vectors of the two functions are $f_1 = \{x_1, x_2, x_3, \cdots, x_n\}$ and $f_2 = \{y_1, y_2, y_3, \cdots, y_n\}$ respectively. Then, their Euclidean distance can be expressed as follows:

$$d = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \tag{6}$$

Table 2: Hyperparameter settings.

Parameter	Value	Parameter	Value
Dimension of embeddings	512	Feed-forward network hidden size	1024
Number of encoder layers	12	Learning rate	0.0001
Number of attention heads	8	Dropout	0.1
Number of graph attention layers	8	Alpha	0.2
Number of graph attention layers	8	Epochs	100

The second one is the task of obfuscating options classification. We add a linear mapping on the basis of the original pretraining model and map the semantic embedding vector obtained previously to the obfuscating option label category. After fine-tuning the model, we use this model to classify the obfuscation options of the binary code.

4.3.1. Binary Function Similarity Comparison Task. In the binary code similarity comparison task, we evaluate SROBR according to the similarity ranking.

We conduct experiments with three obfuscation options (sub, fla, bcf) separately. In each experiment, we perform 1000 rounds of the previously mentioned comparison experiments to ensure the stability and robustness of the model. Through the model, we are able to gain function vectors representing its semantic information. By comparing the vector of the target function with each function vector in the search set, we can rank the functions in the search set according to the similarity. Here, we use $p@n$ to measure the accuracy of the model. The $p@n$ represents the probability that the target function ranks n in the search function set. In particular, we take n as 1, 3, and 5 to measure the pros and cons of the model.

In our experiments, we choose Asm2Vec [6] and SAFE [7] as our benchmark models. Tables 3, 4, and 5, respectively, correspond to the similarity comparison results of the obfuscated binary functions and normal binary functions. From the results, we can find that SROBR performs significantly better than SAFE and Asm2Vec in most cases. Howerver, in the *bcf* option, the Asm2Vec model is slightly better than SROBR, which may be due to the introduction of false basic blocks in the fake control flow confusion technology, which has an impact on the GAT module and causes a slight decrease in accuracy, the random walk algorithm in Asm2Vec just has a certain resistance to it.

4.3.2. Obfuscation Option Classification Task. In the task of obfuscation options classification, we use the previously trained model as a pretraining model and add a linear mapping on this basis, as our classification model:

$$\text{Classify} = \text{MLP}\left(\text{embedding}_f\right), \tag{7}$$

where embedding_f represents the embedding vector obtained by the pretraining model of the function f.

Then, we can fine-tune it by using our labeled dataset. Experiments have proved that high accuracy can be achieved after slight training.

Table 3: sub obfuscation option.

Model	p@1	p@3	p@5
SAFE	0.255	0.427	0.540
Asm2Vec	0.824	0.950	0.977
SROBR	0.903	0.980	0.993

Table 4: bcf obfuscation option.

Model	p@1	p@3	s p@5
SAFE	0.117	0.241	0.337
Asm2Vec	0.802	0.912	0.949
SROBR	0.701	0.878	0.982

Table 5: fla obfuscation option.

Model	p@1	p@3	p@5
SAFE	0.105	0.240	0.341
Asm2Vec	0.165	0.279	0.357
SROBR	0.690	0.881	0.940

We perform four classification tasks on four options. The training effect of the classification model is shown in Figure 11. From this figure, we can see that after 12 epochs of training, the classification accuracy has reached more than 95%. When the training reaches 20 rounds, the accuracy gradually stabilizes at around 98.7%. From Table 6, we can observe the accuracy, recall, and f1-score for each classification option.

From Table 6, we can find that the classification model achieves satisfactory results, which has been pretrained previously. From the perspectives of accuracy, recall, and f1-score, our model is able to capture the internal features of different obfuscation techniques well for accurate identification.

4.4. Ablation Experiments. For the sake of exploring the contribution of each part in SROBR to the overall model framework, we perform extensive ablation experiments, replacing the BERT or GAT module with other layers in the model framework, respectively. Then, we conduct the same training for each model variant and compare the model effects based on the results to analyze the role of each module.

Specifically, we will replace BERT module with linear layer or RNN models and replace GAT module with other

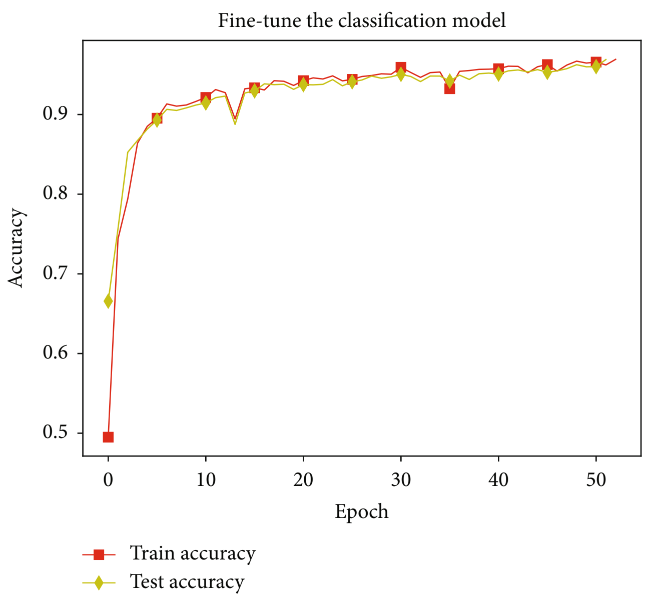


Figure 11: Fine-tune the classification model.

Table 6: Classification results for three obfuscation options.

	Precision	Recall	f1-score
Non	0.97	0.98	0.95
Sub	0.98	0.95	0.96
Fla	1.00	1.00	1.00
Bcf	1.00	0.99	1.00

Table 7: Experimental results of model variants.

options accuracy Variants	*sub*	*bcf*	*fla*
Linear+GAT	0.785	0.675	0.651
LSTM+GAT	0.759	0.577	0.531
BiLSTM+GAT	0.838	0.676	0.682
BERT+GCN	0.852	0.608	0.648
SROBR	0.888	0.701	0.694

Linear+GAT removes the BERT module, and the instruction vector is embedded only by random initialization. It is used to explore the role of the BERT module.
LSTM+GAT uses LSTM instead of BERT to generate the instruction embeddings in the basic block, which is used to compare the effects of BERT and RNN.
BiLSTM+GAT uses BiLSTM to further explore the pros and cons of recurrent neural networks and BERT.
BERT+GCN replaces the GAT module with GCN without using attention weights.

graph neural networks. Since LSTM excels in multiple domains [26–28], we choose it as a comparative experiment.

For all variant models, we train them in the same way. Then, we use the trained model to evaluate on the test set. For each variant, we use three obfuscation options, with the same method in Section 4.3.1 to compare its accuracy according to p@1. The results are shown in Table 7. From the results, we can clearly see that in terms of antialiasing ability, the BERT model is significantly better than the linear model and the recurrent neural network, and the GAT module is better than GCN.

5. Conclusion

In this paper, we propose a novel neural network structure SROBR for obfuscated code to obtain the semantic embedding representation of binary functions. It mainly contains three submodules. The first submodule uses BERT module to capture sequence information to generate instruction embeddings. The second submodule aggregates the instruction embeddings in basic block through FFN to obtain the embedding of the basic block. The third submodule gets the structural information of the function through basic blocks and the adjacency matrix, thereby obtaining the semantic vector of the entire function.

Through extensive comparative experiments on our dataset, we have proved that the proposed model can better deal with the obfuscation options of O-LLVM. However, other obfuscation options are not used for verification, which can be further done as our future research direction.

Data Availability

Our dataset can be available from the GitHub repositories, which mainly includes 9 projects, such as OpenSSL (https://github.com/openssl/openssl), libGmp (https://github.com/sethtroisi/libgmp), libTomCrypt (https://github.com/libtom/libtomcrypt), SQLite (https://github.com/sqlite/sqlite), Busybox (https://github.com/mirror/busybox), Coreutils (https://github.com/coreutils/coreutils), Diffutils (https://www.gnu.org/software/diffutils/), Libcurl (https://github.com/curl/curl), and Zlib (https://github.com/madler/zlib).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by Natural Science Foundation of China under Grant no. 61802435.

References

[1] I. U. Haq and J. Caballero, “A survey of binary code similarity,” *ACM Computing Surveys*, vol. 54, no. 3, 2021.

[2] Q. Feng, R. Zhou, C. Xu, Y. Cheng, B. Testa, and H. Yin, “Scalable graph-based bug search for firmware images,” in *Proceedings of the ACM Conference on Computer and Communications Security*, Vienna, Austria, 2016.

[3] X. Xu, C. Liu, Q. Feng, H. Yin, L. Song, and D. Song, “Neural network- based graph embedding for cross-platform binary code similarity detection,” in *Proceedings of the ACM Conference on Computer and Communica- tions Security*, pp. 363–376, Dallas Texas USA, 2017.

[4] J. Gao, X. Yang, Y. Fu, Y. Jiang, and J. Sun, “Vulseeker: a semantic learning based vulnerability seeker for cross-platform binary,” in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pp. 896–899, Montpellier France, 2018.

[5] F. Zuo, X. Li, P. Young, L. Luo, Q. Zeng, and Z. Zhang, “Neural machine translation inspired binary code similarity comparison beyond function pairs,” in *Proceedings 2019 Network and Distributed System Security Symposium*, San Diego, California, 2019.

[6] S. H. H. Ding, B. C. M. Fung, and P. Charland, “Asm2Vec: boosting static representation robustness for binary clone search against code obfuscation and compiler optimization,” in *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 472–489, San Diego, California, 2019.

[7] L. Massarelli, G. A. Di Luna, F. Petroni, R. Baldoni, and L. Querzoni, “SAFE: self-attentive function embeddings for binary similarity,” in *Detection of Intrusions and Malware, and Vulnerability Assessment. DIMVA 2019*, R. Perdisci, C. Maurice, G. Giacinto, and M. Almgren, Eds., vol. 11543 of Lecture Notes in Computer Science, pp. 309–329, Springer, Cham, 2019.

[8] X. Zhang, W. Sun, J. Pang, F. Liu, and Z. Ma, “Similarity metric method for binary basic blocks of cross-instruction set architecture,” in *Proceedings 2020 Workshop on Binary Analysis Research*, San Diego, California, 2020.

[9] J. Yang, C. Fu, X. Y. Liu, H. Yin, and P. Zhou, “Codee: a tensor embedding scheme for binary code search,” *IEEE Transactions on Software Engineering*, p. 1, 2021.

[10] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Proceedings of the 26th International Conference on Neural Informa- tion Processing Systems - Volume 2. NIPS'13*, pp. 3111–3119, Red Hook, NY, USA, 2013.

[11] Z. Yu, R. Cao, Q. Tang, S. Nie, J. Huang, and S. Wu, “Order matters: semantic-aware neural networks for binary code similarity detection,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1145–1152, New York, 2020.

[12] H. Koo, S. Park, D. Choi, and T. Kim, “Semantic-aware binary code representation with bert,” 2021, https://arxiv.org/abs/2106.05478.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” 2018, https://arxiv.org/abs/1810.04805.

[14] D. Peng, S. Zheng, Y. Li, G. Ke, D. He, and T.-Y. Liu, “How could neural networks understand programs?,” in *International Conference on Machine Learning*, pp. 8476–8486, 2021.

[15] A. Vaswani, N. Shazeer, N. Parmar et al., “Attention is all you need,” *Advances in Neural Information Processing Systems*, pp. 5999–6009, 2017.

[16] J. Ming, F. Zhang, D. Wu, P. Liu, and S. Zhu, “Deviation-based obfuscation-resilient program equivalence checking with application to software plagiarism detection,” *IEEE Transactions on Reliability*, vol. 65, no. 4, pp. 1647–1664, 2016.

[17] S. Alrabaee, P. Shirani, L. Wang, and M. Debbabi, “FOSSIL: a resilient and e_cient system for identifying FOSS functions in malware binaries,” *ACM Transactions on Privacy and Security*, vol. 21, no. 2, pp. 1–34, 2018.

[18] P. Junod, J. Rinaldini, J. Wehrli, and J. Michielin, “Obfuscator-LLVM – software protection for the masses,” in *2015 IEEE/ACM 1st International Workshop on Software Protection*, pp. 3–9, Florence, Italy, 2015.

[19] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” *Advances in neural information processing systems*, vol. 26, pp. 1310–4546, 2013.

[20] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” 2016, https://arxiv.org/abs/1609.02907.

[21] Y. Huang, M. Qiao, F. Liu, X. Li, H. Gui, and C. Zhang, “Binary code traceability of multigranularity information fusion from the perspective of software genes,” *Computers & Security*, vol. 114, article 102607, 2022.

[22] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” 2018, https://arxiv.org/abs/1710.10903.

[23] W. L. Hamilton, R. Ying, and J. Leskovec, “Representation learning on graphs: methods and applications,” 2017, https://arxiv.org/abs/1709.05584.

[24] M. Qiao, X. Zhang, H. Sun et al., “Multi-level cross-architecture binary code similarity metric,” *Arabian Journal for Science and Engineering*, vol. 46, no. 9, pp. 8603–8615, 2021.

[25] L. Massarelli, G. A. Di Luna, F. Petroni, L. Querzoni, and R. Baldoni, “Investigating graph embedding neural networks with unsupervised features extraction for binary analysis,” in *Proceedings 2019 Workshop on Binary Analysis Research*, pp. 21–24, San Diego, California, 2019.

[26] L. Lv, Z. Wu, J. Zhang, Z. Tan, L. Zhang, and Z. Tian, “A vmd and lstm based hybrid model of load forecasting for power grid security,” *IEEE Transactions on Industrial Informatics*, p. 1, 2021.

[27] L. Zhang, C. Xu, Y. Gao, Y. Han, X. Du, and Z. Tian, “Improved dota2 lineup recommendation model based on a bidirectional lstm,” *Tsinghua Science and Technology*, vol. 25, pp. 712–720, 2020.

[28] L. Zhang, Z. Huang, W. Liu, Z. Guo, and Z. Zhang, “Weather radar echo prediction method based on convolution neural network and long short- term memory networks for sustainable e-agriculture,” *Journal of Cleaner Production*, vol. 298, 2021.

Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 8641220, 12 pages
https://doi.org/10.1155/2022/8641220

Research Article

NC-GNN: Consistent Neighbors of Nodes Help More in Graph Neural Networks

Ming Xu, Baoming Zhang, Hualei Yu, Jinliang Yuan, and Chongjun Wang

National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China

Correspondence should be addressed to Chongjun Wang; chjwang@nju.edu.cn

Received 24 March 2022; Accepted 28 May 2022; Published 18 June 2022

Academic Editor: Lei Zhang

Graph neural networks, as the promising methodology in data mining for graph data, currently attract much attention and are broadly applied in graph-based tasks. Existing GNN methods mostly follow the assumption of homophily, where the connected nodes are similar and share the same labels. Most graphs in the real world can satisfy the assumption. However, for the particular nodes, the situation is not always satisfied. The connections between different-labeled nodes will introduce noise in feature aggregation and result in node representation deviating in the wrong direction. In this paper, we focus on the different-labeled neighbors of labeled nodes in the graphs. By regarding aggregation among neighbors as the procedure of node feature reconstruction, we devise a novel metric *neighbor consistency* to measure the difference between nodes and their neighborhoods. In this way, we can evaluate the reliability of nodes after aggregation. Furthermore, we propose a novel method, *Neighbor Consistent Graph Neural Networks* (NC-GNN), to promote the training of graph neural networks by reweighting the influence of labeled nodes based on neighbor consistency scores. Systematic experiments are conducted on benchmark datasets, and the results demonstrate the effectiveness of our method.

1. Introduction

Graphs are widely observed in our daily lives, as graphs can well capture objects' features and the abundant interactions between objects. For instance, following-relations in users form friendship graphs in social networks [1], users' interactions on goods construct user-item graphs in recommendation systems [2], and the communications between mobile phones construct graphs in cellular networks [3]. As an important part of data mining, graph learning attracts much attention to learning latent information in real-world graph data recently. Graph neural networks promote graph learning by introducing deep learning frameworks and achieve great success in most graph mining tasks, like node classification [4, 5], link prediction [6, 7], and graph classification [8–10]. The frameworks are also widely applied in real-life graph tasks, from recommendation [11, 12], social networks [1], text extraction [13, 14], knowledge graphs [15], etc.

The goal of graph neural networks is to encode nodes in the graph into dense and low-dimensional embeddings with node features and graph topology information preserved simultaneously. In this way, nodes or graphs can be represented by the embeddings, and then, we can find out the latent information for downstream tasks. Existing graph neural frameworks mostly follow the manner of message passing neural network (MPNN) [16], namely, updating nodes' representations by aggregating information from neighborhoods. In this way, the representation of nodes is smoothed in each iteration, and the final representation can be used for downstream tasks. The most popular model, GCN [4], simplifies the message passing strategy by using the first-order polynomial, namely, only considering the direct neighbors of nodes. A lot of variants of GCN are then proposed [5, 17].

As graphs in the real world contain abundant nodes, existing graph neural networks primarily focus on the semi-supervised scenario where partial nodes are tagged with labels. MPNN-based GNNs achieve great success by following the assumption of homophily, which assumes that connecting nodes in the graph are similar in features and share the same labels. In this way, beneficial information will be aggregated to nodes and can help models learn better representations. Most graphs in the real world can satisfy the assumption, while for the particular node in the graph, the situation is not always satisfied. For instance, there exist mistakes when collecting graph data that link nodes of differently labeled nodes. The adversarial attack in graphs is often conducted by connecting nodes of different classes as message passing will amplify noise from different-labeled nodes. Besides, nodes in the boundaries between classes connect to different-labeled nodes in the connected graph. We did simple statistics on benchmark datasets by counting the nodes with different-labeled neighbors. We call the nodes *neighbor inconsistent nodes* (NI-Nodes). The results are shown in Table 1. From the table, we can figure out that NI-Nodes are common even in widely used homophily graphs.

When performing message passing in these nodes, unnecessary information will be aggregated. Consequently, the final node representation will deviate in the wrong direction. Here is a toy example:

In Figure 1, we conducted 1-step aggregation in the sample graph, and different colors indicate different labels. According to the figure, as the node a and node d share the consistent neighbors, respectively, their colors remain the same after aggregation. However, node b and node c change their colors with different-colored neighbors' information propagated, resulting in unreliable final representations. What is worse, the noisy information will influence other nodes with the procedure of iterative aggregation.

Consequently, how to evaluate messages from neighbors becomes crucial for better graph neural networks. The most popular method, GCN, treats every neighbor node equally by mean aggregator without considering noise from the neighborhood. GAT [5] introduces attention scores to evaluate the influence of every neighbor and then reweight messages from the neighborhood. Nevertheless, it still assumes that all neighbors are beneficial for the model training. Some other methods [18, 19] also try to modify the topology structure to help better aggregate information. Besides, some self-supervised learning methods [20–23] try to construct multiview graphs and apply contrastive learning to alleviating the noise from unreliable neighbors. However, the methods usually train multiple models for different graph views and learn multiple representations for nodes, which is time and space consuming.

In this paper, we focus on different-labeled neighbors in the aggregation of nodes. Instead of modifying models from the structure or neighbor weights, we attempt to consider this problem from the point of labeled nodes as the framework is optimized by the nodes. We argue that considering aggregation in graph neural networks can be seen as the procedure of node feature reconstruction. And we can divide the information captured in aggregation into two parts, node features and context features. Then, we measure the difference between the node feature and the context feature to evaluate the reliability of node representation after aggregation. In particular, we devise a novel metric NC (neighbor consistency) to evaluate the reliability. Furthermore, we propose the method called *Neighbor Consistent Graph Neural Networks* (NC-GNN) to improve the training of graph neural networks by reweighting the influence of labeled nodes. The greater the neighbor consistency is, the more reliable is the node representation after aggregation, which indicates that the node representation can help more in the model training and vice versa. Empirical results for node classification demonstrate the effectiveness of our method. We summarize the main contributions of this paper as follows:

Table 1: Nodes with different-labeled neighbors in the benchmark graphs.

	Nodes	NI-Nodes	Proportion (%)
Cora	2078	932	34.42
Citeseer	3327	1360	40.88
PubMed	19717	8759	35.57

(i) We devise neighbor consistency (NC) to measure the difference between labeled nodes and their neighborhoods. By regarding aggregating information from neighborhoods as node feature reconstruction, NC can evaluate the reliability of labeled nodes after aggregation effectively

(ii) We devise a novel method NC-GNN to promote the training process of graph neural networks. The method can obtain better embeddings from neighbor-consistent nodes by reweighting the influence of labeled nodes according to neighbor consistency scores

(iii) We conduct extensive experiments on node classification, and the results indicate the effectiveness of our method

The remaining part of the paper is organized as follows. Section 2 reviews the related works involving graph neural networks and some modifications of aggregation. In Section 3, we introduce some preliminaries and the framework of graph neural networks. In Section 4, our method is then presented with a detailed description. Extensive experiments are conducted in Section 5 to evaluate the performance of our method. At last, Section 6 concludes the paper with discussions and future works.

2. Related Works

In this section, we briefly review the related works, including graph neural networks and modifications for aggregating neighbor nodes in graph neural networks. Since graph neural network is a very active research area, we only introduce the most relevant models. For more details, we refer readers to some surveys [24, 25].

2.1. Graph Neural Networks. The research of graph neural networks is popular in graph learning. It is aimed at transferring traditional convolutional networks from Euclidean space to graph domain. Graph convolution is first proposed in [26] in graph signal processing, and there are many works to simplify the framework in both spectral and spatial

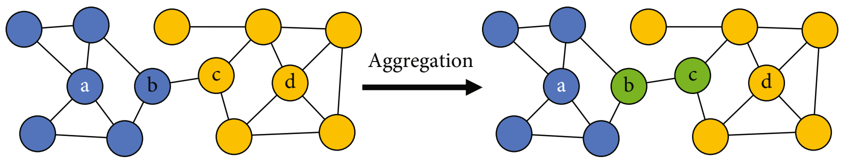


FIGURE 1: An example of 1-step aggregation in graph neural networks.

domain. For example, [27] introduces the Chebyshev polynomials with orders of K to approximate the eigendecomposition. And Kipf and Welling [4] in GCN simplify the model by using the first-order polynomial, namely, only considering the direct neighbors of nodes. Due to the simpleness and conciseness of GCN, it becomes the baseline and popular in graph learning. Existing graph neural network models usually follow the framework of MPNN (message passing neural network) [16], which aggregates messages from neighbor nodes to update the embeddings of target nodes. For instance, GraphSage [17] applies different strategies to aggregate features from neighbor nodes. GAT [5] evaluates the importance between target nodes and neighbor nodes so that the model can aggregate more related information. GIN [9] develops a simple structure to ensure that the aggregator is injective and the representational power is equal to the power of WL-test. SGCN [28] simplifies GCN through successively removing nonlinearities and collapsing weight matrices between consecutive layers.

The models based on MPNN mostly follow the assumption of homophily, which states that nodes connected by edges are similar and beneficial information can be propagated in the graph. However, the assumption is not always satisfied as there always exists unintentional or intentional noise in real-world graphs. In the following subsection, we will introduce some modifications on aggregating neighbor nodes considering the situation.

2.2. Modifications on Aggregating Neighbor Nodes. As homophily is not always satisfied in the real world, aggregating beneficial neighbors becomes crucial in graph neural networks. To alleviate the influence of different-labeled neighbors, many works are then proposed. For instance, [29] compares the original prediction with the counterfactual prediction calculated by presenting multiple data indicators to assess the trustworthiness of neighbor nodes. Besides, as neighborhood information is preserved in the graph structure, many frameworks are then designed to modify the graph structure so as to conduct aggregation better. Methods like self-enhanced GNN [30] and EGAI [31] add or remove edges based on the predicted neighbor labels learned by the model. Bayesian GCN [32], LDS [18], SimP-GCN [33], and IDGL [34] adopt different strategies to optimize the graph structure and node embeddings simultaneously to make graph structure more suitable for model learning. Some contrasting models try to construct multiviews by modifying neighbor structures [35–37]. In the real-world scenario, some models adopt neighbor aggregation modifications to better fit downstream graph tasks, like modifying graph structure [20, 38] or evaluating the dependencies between nodes [39, 40].

Though the above methods achieve great progress in encoding nodes into better embeddings with modified structure, the modification of graph structure sometimes discards the important interactions between nodes, resulting in information loss.

3. Background

3.1. Notations and Preliminaries. This paper mainly focuses on undirected graphs, but the method can also be used in directed graphs. We present $\mathscr{G}=(\mathscr{V},\mathscr{E},\mathscr{X})$ as a graph, where $\mathscr{V}$ consists of the set of nodes in $\mathscr{G}$, with $|\mathscr{V}|=N$. $\mathscr{E}$ is the collection of edges. $\mathscr{X}\in\mathbb{R}^{N\times F}$ denotes node feature matrix, where $\mathbf{x_i}\in\mathbb{R}^F$ represents the attributes of node v_i, and F is the dimension of node features. Adjacency matrix $\mathbf{A}\in\{0,1\}^{n\times n}$ is the topological structure of graph $\mathscr{G}$, where $\mathbf{A}_{ij}>0$ indicates that there is an edge between nodes i and j. Otherwise, $\mathbf{A}_{ij}=0$.

Given topological structure $\mathbf{A}$ and feature matrix $\mathscr{X}$ as input, our objective is to learn low-dimensional dense node embedding matrix $\mathbf{Z}\in\mathbb{R}^{N\times d}$ with $d\ll F$ without human annotation. The learned node embeddings can well preserve topology and feature information so as to be applied to downstream tasks. In this paper, we focus on semisupervised node classification. $\mathscr{V}_L\subset\mathscr{V}$ is the labeled node set, and we have $|\mathscr{V}_L|\ll|\mathscr{V}|$. $\mathscr{Y}\in\mathbb{R}^{N\times k}$ is the label set for nodes in the graph, $y_i\in\mathbb{R}^k$ is the one-hot label vector of node v_i, and k is the number of classes. Then, we aim to train a classifier $\widehat{\mathscr{Y}}_U=g(\cdot)$ by utilizing the learned node embedding $\mathbf{Z}$ as input to predict the labels for unlabeled node set $\mathscr{V}_U=\mathscr{V}-\mathscr{V}_L$.

3.2. Graph Neural Networks. Graph neural networks are a popular class of graph embedding methodologies that model the graph structure and node features to encode representations for nodes in the graph. Existing GNN frameworks mostly learn node representations by aggregating the features of neighbor nodes. The output of the k-th layer of the framework can be generally expressed as

$$h_i^{(k)}=\sigma\left(h_i^{(k-1)},\mathrm{AGG}\left(h_j^{(k-1)}\right)\right),j\in\mathscr{N}(i),\tag{1}$$

where $h_i^{(k)}$ is the node representation of node v_i at the k-th layer with $h_i^{(0)}=\mathbf{x}_i$ and $\mathscr{N}(i)$ is the direct neighbors of node v_i. $\sigma(\cdot)$ is the nonlinear method to combine the information from the previous layer to update node representations. $\mathrm{AGG}(\cdot)$ is the method to aggregate information from

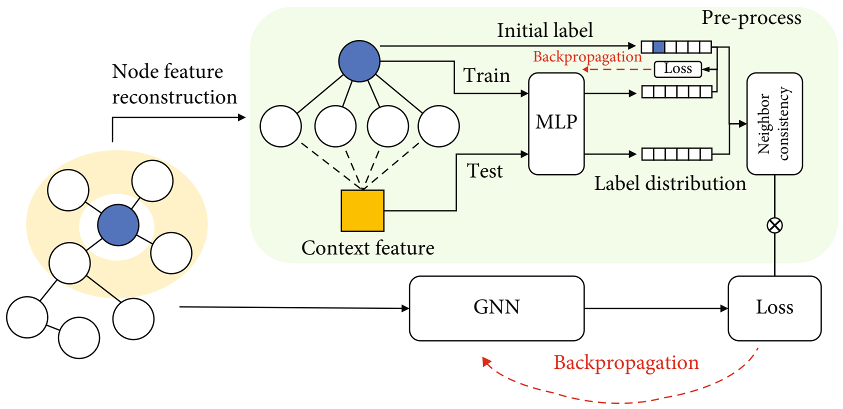


Figure 2: Overview of our proposed NC-GNN. The blue-colored node represents labeled node. We evaluate neighbor consistency of labeled nodes in preprocessing section. After that, we reweight the influence of labeled nodes in the loss function with neighbor consistency scores.

neighbor nodes, which is usually mean, max, sum methods. Different GNN method varies in the formulations of $\sigma(\cdot)$ and $\mathrm{AGG}(\cdot)$ methods.

4. Our Approach

We propose a method called Neighbor Consistent Graph Neural Networks, short as NC-GNN, to promote the training of GNNs by evaluating the consistency between nodes and the corresponding neighbors. The overview of the method is shown in Figure 2. The model consists of two components. In the preprocessing procedure, we evaluate neighbor consistency for labeled nodes. After that, with the calculated neighbor consistency scores, the influence of labeled nodes is reweighted in the model training procedure. We will introduce each component in detail as follows.

4.1. Node Feature Reconstruction. Graph neural networks essentially utilize the message passing strategy of aggregating information from neighbor nodes to update node representations. For every node, the desiring situation is that connected neighbor nodes are all similar, namely, the assumption of homophily, so that the final node representation can be more accurate and generalized for node classification. However, the hypothesis is not always satisfied for particular nodes as discussed above. Therefore, the information from neighbor nodes should be evaluated before messaging passing.

Based on the framework of GNNs shown in Section 3.2, the update of the representation for nodes can be divided into node features and context features, which preserve all the neighbor features. As node feature remains unchanged, GNN is trying to transform node representation towards context features. With infinite iterations of aggregation and update, the node representation can be transferred into the mixture feature constructed by neighbors. If node features are blank, the final representation of the node is then decided by the context features thoroughly. Based on the above discussion, we can regard the information aggregation as node feature reconstruction, which represents the nodes with the weighted combination of context features from neighbor nodes and node features.

Therefore, the context features play a critical impact in the final embeddings of nodes after aggregation. In particular, to ensure the final embeddings can well represent the nodes, the context feature constructed by neighbor nodes should be similar to the node features. And the corresponding labels of context features can well capture the information.

In the Euclidian space, we assume a virtual center node exists for every class. Therefore, if neighbor nodes are similar to the target node and share the same label, the nonnegative weighted sum of neighbor features, namely, context features, should always be closer to the class center node than the neighbor node which is furthest away. Otherwise, the label of the representation after aggregation cannot be guaranteed.

Consequently, we devised a simple metric, called neighbor consistency, to better evaluate the consistency between nodes and the corresponding context features by measuring the difference between the labels of nodes and their neighbors.

4.2. Evaluation of Neighbor Consistency. To help GNNs learn better node embeddings, nodes with consistent context representations should be more important. When the consistency is high, the final embedding of the node is representative. Firstly, we calculate the context feature as

$$c_i = \sum \alpha_{i,j} \mathbf{x}_j, j \in \mathcal{N}(i), \tag{2}$$

where $\mathcal{N}(i)$ is the neighbor set of v_i. Actually, the equation can be seen as encoding the neighborhood of v_i to a virtual context node. $\alpha_{i,j}$ is the weight between v_i and v_j. If we focus on the direct neighbors in the graph just as the aggregation in GCN, we can simply set $\alpha_{i,j} = 1/|\mathcal{N}(i)|$. Or we can construct the ego-network of v_i with limited k-order neighbors and then calculate $\alpha_{i,j}$ through Personalized PageRank. In this way, we can reconstruct node features through a wider receptive field.

There exist many methods to measure the difference between features. In this paper, as we focus on labeled nodes, we propose the Multilayer Perception (MLP) to classify context features. In particular, we regard labeled nodes as the training set for the classifier, and then, we can get the following label distribution for the corresponding context features, namely,

$$\widehat{\mathbf{y}}_i^c = \mathrm{MLP}(c_i), \tag{3}$$

where $\widehat{\mathbf{y}}_i^c \in \mathbb{R}^k$ is the predicted label distribution for context feature of node v_i and k represents the number of classes.

Compared with other methods like Euclidian distance, $\mathrm{MLP}(\cdot)$ trained by labeled nodes can better capture class information among all the training samples, so the $\mathrm{MLP}(\cdot)$ is more generalized to calculate robust label distributions for context features.

The learned $\mathrm{MLP}(\cdot)$ is overfitting with labeled nodes. So it can well classify the context features. While the model cannot perform well in unlabeled nodes, that is the reason why we do not consider evaluating neighbor consistency for unlabeled nodes in the whole graph.

$\widehat{\mathscr{Y}}_L^c$ can then be used to evaluate neighbor consistency by comparing with labels $\mathscr{Y}_L$ of corresponding labeled nodes. When the context feature's predicted label is the same as the corresponding labeled node, we can conclude that the labeled node is neighbor-consistent. The higher confidence in the prediction indicates the greater consistency between the node and the neighborhoods. Otherwise, if the context feature is classified as a different label, the neighbors are inconsistent with the labeled node. In this paper, we utilize prediction confidence to evaluate the consistency between neighbors and nodes. So we define neighbor consistency score (NC) as

$$\mathrm{NC}_i = \begin{cases} \max\left(\left\{\widehat{\mathbf{y}}_{ij}^c\right\}\right), & \mathrm{argmax}_j\left(\left\{\widehat{\mathbf{y}}_{ij}^c\right\}\right) = l, \mathbf{y}_{il} = 1, \\ -1 * \max\left(\left\{\widehat{\mathbf{y}}_{ij}^c\right\}\right), & \text{otherwise}, \end{cases} \tag{4}$$

where max $(\cdot)$ function is used to identify the maximum value in the label distribution. $\mathrm{argmax}(\cdot)$ is to figure out the class label of the context feature with the highest probability.

The calculated NC scores can capture the consistency between labeled nodes and their neighbors well, with larger values indicating greater consistency.

4.3. Promoting GNN by Loss Weight Reweighting. In this section, we introduce NC-GNN, a training weight schedule mechanism to promote the training of graph neural networks. As discussed above, graph neural networks are trained through aggregating neighbor features for target nodes to update node representations iteratively, and the model is optimized by calculating the classification loss of labeled nodes. As a result, the labeled nodes with consistent neighborhoods can help more in the training process. Based on the calculation of neighbor consistency, we can figure out the labeled nodes whose neighbors' features may not be helpful in aggregation and even bring extra noise. So we can utilize NC scores to make nodes with consistent neighbors play a more active role in model learning. Specially, we devise a simple method for the calculation of node weights according to neighbor consistency scores,

$$W = \mathrm{softmax}(\mu \cdot \mathrm{NC} + \mathbb{I}(\mathrm{NC}) \cdot b), \tag{5}$$

$$\mathbb{I}(x) = \begin{cases} 1, & x > 0, \\ -1, & x < 0, \end{cases} \tag{6}$$

where $W \in \mathbb{R}^L$ are the weight matrix for labeled nodes and μ and b are the parameters of scaling the neighbor consistency scores and initial weight for nodes, respectively. $\mathbb{I}(\cdot)$ is the indicator function to give the initial weight difference between neighbor-consistent nodes and neighbor-inconsistent nodes. Here we choose $\mathrm{softmax}(\cdot)$ to avoid future normalization in calculation.

Then, the training loss for a better graph neural network is computed by the following equations,

$$\widehat{\mathscr{Y}} = \mathrm{softmax}(\mathbb{F}(\mathscr{X}, \mathbf{A}, \theta)), \tag{7}$$

$$L_N = -\frac{1}{|\mathscr{L}|}\sum_{i\in\mathscr{L}} w_i \sum_{l=1}^{k} \mathbf{y}_{il} \log \widehat{\mathbf{y}}_{il}, \tag{8}$$

where $\mathbb{F}$ denotes any GNN framework, θ is the parameter of $\mathbb{F}$, $\widehat{\mathscr{Y}}$ is the GNN output. w_i is the loss weight of labeled node v_i calculated in Equation (8), $\widehat{\mathbf{y}}_i$ is the prediction of v_i, and $\mathbf{y}_i$ indicates the original label for node v_i in one-hot embedding. By encouraging positive effects from the aggregation of consistent neighbors which share the same label with target nodes and alleviating the negative effects from the aggregation of neighbors with different labels, our method can promote graph neural networks by reducing noise in model training, so as to represent nodes with robust embeddings.

4.4. Complexity Analysis. In our NC-GNN method, we construct context representations for labeled nodes and predict the corresponding label distribution to evaluate the neighbor consistency between labeled nodes and their neighbors. And the method can be split into two procedures, preprocessing and GNN training.

In the GNN training procedure, we use common GNN frameworks and the time complexity is the same as the frameworks. Here we consider GCN as an example. The time complexity of an L-Layer GCN model is $O(L|\mathbf{A}_0|F + L NF^2)$, where N is the number of nodes, $|\mathbf{A}_0|$ is the number of nonzeros in $\mathbf{A}$, and F is the number of features. In the preprocessing procedure, we first construct the context representations for labeled nodes. For the mean method, the time complexity is $O(d|\mathscr{L}|)$, where d is the average degree of nodes in the graph and $|\mathscr{L}|$ is the number of labeled nodes. For the PPR method, the complexity is $O(d^2|\mathscr{L}|)$. Due to the sparsity of graphs, $d < <N$. Then, we use $\mathrm{MLP}(\cdot)$ as the classifier to predict the contexts' label distributions. For the L-Layer network, the time complexity is $O(L|\mathscr{L}|F^2)$. And the overall complexity of the procedure is $O(|\mathscr{L}| + L|\mathscr{L}|F^2)$.

For space complexity, GCN needs $O(LF^2)$ memory for storing the weight matrix and $O(LNF)$ for embeddings.

Table 2: Data distribution of benchmark datasets.

	Cora	Citeseer	PubMed	Photo	Computers
Nodes	2078	3327	19717	7487	13381
Edges	5278	4614	44325	119043	245778
Features	1433	3703	500	745	767
Classes	7	6	3	8	10
Training nodes	140	120	60	20 per class	20 per class
Validation nodes	500	500	500	30 per class	30 per class
Test nodes	1000	1000	1000	Rest nodes	Rest nodes

Our method needs additional $O(L|\mathscr{L}|F)$ memory to store the context representations and $O(LF^2)$ memory to store the weight matrix of the classifier.

5. Experiments

In this section, we conduct adequate experiments to validate the effectiveness of our method. We first evaluate whether the calculated neighbor consistency matches the neighbor distribution in the graphs. Then, node classification experiments are conducted to demonstrate the effectiveness of our method. Furthermore, we discuss the relationships between neighbor consistency and the performance of our method and study the parameters' influence on the model.

5.1. Datasets. Following previous works [4, 41], we utilize the widely used Planetoid paper citation datasets(Cora, Citeseer, and PubMed) and the Amazon purchase graphs (Photo and Computers). In the citation datasets, nodes and edges represent documents and citation relations between documents, respectively. Each node is represented by the bag-of-words features extracted from the contents of the document. Each node corresponds to a label with the one-hot encoding of the document category. In Amazon purchase graphs, nodes represent goods on the site, edges indicate that two goods are frequently bought together, node features are bag-of-words encoded product reviews, and class labels are given by the product category. We employ data with *DGL* [42] and *Pytorch-Geometric* [43] module, and the data distribution is shown in Table 2.

5.2. Experimental Settings

5.2.1. Baseline Methods. To evaluate the effectiveness of our method, we compare with the following *state-of-the-art* methods.

(i) *DeepWalk* [44]. It is the typical shallow network embedding model by regarding node as words in documents and utilizing skip-gram models to train embeddings

(ii) *GCN* [4]. It is the baseline of graph neural networks. It generalizes the covolutional operation from deep learning to graph domain, considering aggregating messages from direct neighbors

Table 3: Proportion of NI-Nodes to the predicted nodes using mean method.

	Predicted nodes	NI-Nodes	Percentage
Cora	36	20	55.6%
Citeseer	46	37	80.4%
PubMed	7	6	85.7%
Photo	37	31	83.8%
Computers	67	52	77.6%

Table 4: Proportion of NI-Nodes to the predicted nodes using Personalized PageRank method.

	Predicted nodes	NI-Nodes	Percentage
Cora	28	20	71.4%
Citeseer	38	32	84.2%
PubMed	7	6	85.7%
Photo	34	28	82.4%
Computers	69	58	84.1%

(iii) *GraphSage* [17]. It is extending the mean aggregator of GCN to perform multiaggregation and performing a sampling strategy before aggregation

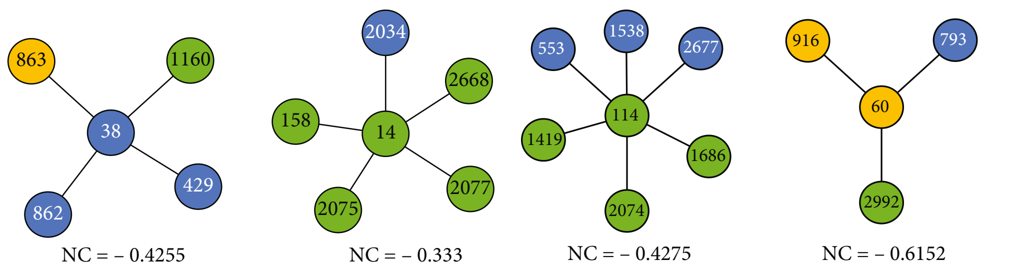
(iv) *GAT* [5]. It is considering weighting the neighbors in aggregation by introducing attention mechanism to GCN and assigning different weights to neighbor nodes according to attention scores

(v) *DropEdge* [45]. It is considering modifying the structure by randomly removing a certain number of edges at each epoch to improve the generalization capacity of GCN

(vi) *SimP-GCN* [33]. It is considering modifying the structure by combining kNN-graph calculated by node features and original graph to preserve node feature similarity with updated structure and improve the homophily in the graph

(vii) *NC-GNN*. This is our method, and we choose the GCN and GAT as our baseline methods

5.2.2. Parameter Settings. In parameter settings, we designed 2-layer graph neural networks with the same hidden layer dimension and the same output dimension simultaneously for every method. For baseline methods like DeepWalk,



(a) Samples in Cora (b) Samples in Citeseer

Figure 3: Sampled predicted low neighbor consistency nodes in datasets. Numbers included are the node index, and different color indicates different class.

GCN, GAT, and DropEdge, we follow the instruction of original codes in Github published by the authors. For GraphSage, we only consider the situation with the mean aggregator, and the model is implemented the same as the authors' guidance. With our methods, we set the parameters of MLP almost the same as GCN, with the same hidden layers, and the same dropout rate. Besides, for the NC-GNN models, we follow most settings the same as the base methods except that we use an early stopping strategy the same as GAT with patience of 100 epochs in NC-GCN.

In data splitting, we follow the same data split as previous works in Planetoid citation datasets. For Amazon copurchasing datasets, as there is no existing split setting, we randomly sample 20 nodes per class as the training set, 30 nodes per class as the validating set, and the rest nodes as the testing set, which is consistent with previous works.

5.3. Neighbor Consistency Evaluation. We first evaluate the neighbor consistency measured by the difference between nodes and the corresponding neighbors. According to the discussion in Section 4.2, we can figure out that when the value of the NC score is negative, the node is more likely to connect to different-labeled neighbor nodes. So we conduct the experiments to determine whether the predicted neighbor-inconsistent nodes are connecting to different-labeled nodes. Besides, we compare the mean and Personalized PageRank method in constructing context features. The results are shown in Tables 3 and 4.

From Tables 3 and 4, we can find out that most nodes with negative NC scores are neighbor-inconsistent, which may bring unnecessary information for aggregation in model training. The results prove that NC scores can well capture the neighbor consistency of nodes. Compared Tables 3 and 4, in most cases, Personalized PageRank method performs better than mean method in finding neighbor-consistent nodes, as the wider receptive field can provide more neighbor information. Therefore, we utilize Personalized PageRank method as the base method for constructing context features in the following experiments.

Visualization. We sampled some predicted nodes by our method in the datasets to observe whether NC scores can figure out neighbor-inconsistent nodes. The results are shown in Figure 3.

From Figure 3, we can conclude that NC scores can discover the neighbor-inconsistent nodes in the graph well. As there exist different-labeled nodes in the neighborhood, the aggregated information can be noise for the central node sometimes. Besides, in the right part of Figure 3(a), we can find that only one noisy node in the neighborhood sometimes can bring a considerable negative impact on aggregation. The results indicate that attention to neighbor consistency is essential to train GNN models.

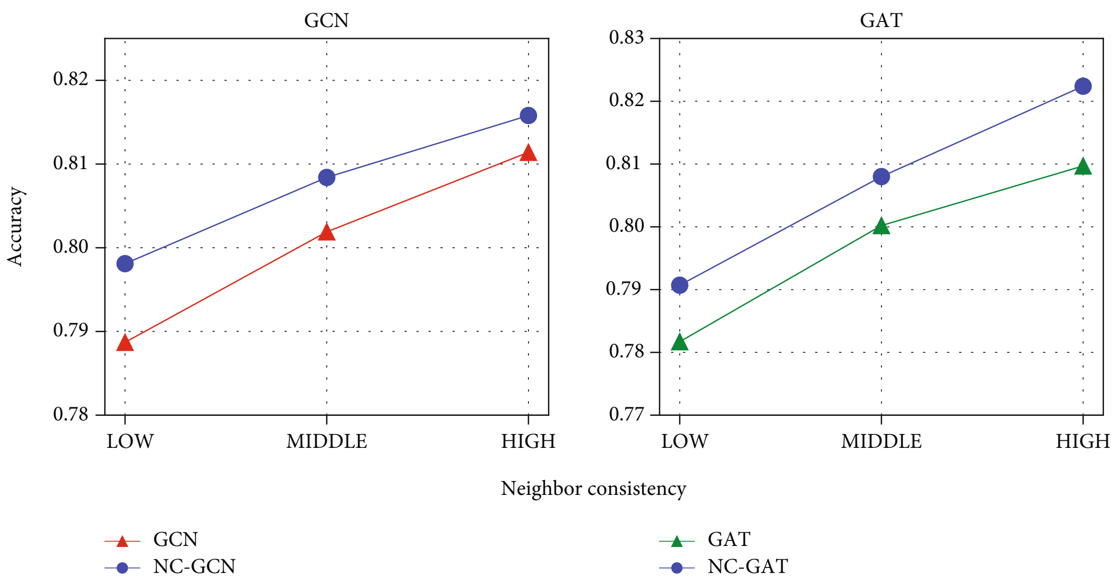
Table 5: The results of node classification accuracy (%) on the datasets.

Data	Cora	Citeseer	PubMed	Photo	Computers
DeepWalk	67.2	43.2	65.3	-	-
GCN	81.6	70.5	78.7	89.6	76.5
GraphSage	77.4	67.0	76.6	86.5	74.5
GAT	82.6	70.3	77.5	91.3	79.3
DropEdge	79.6	67.6	73.4	87.6	78.2
SimP-GCN	82.5	72.5	**80.9**	89.4	78.2
NC-GCN	83.0	**74.2**	79.3	90.9	83.3
NC-GAT	**83.5**	73.0	79.1	**92.1**	**84.5**

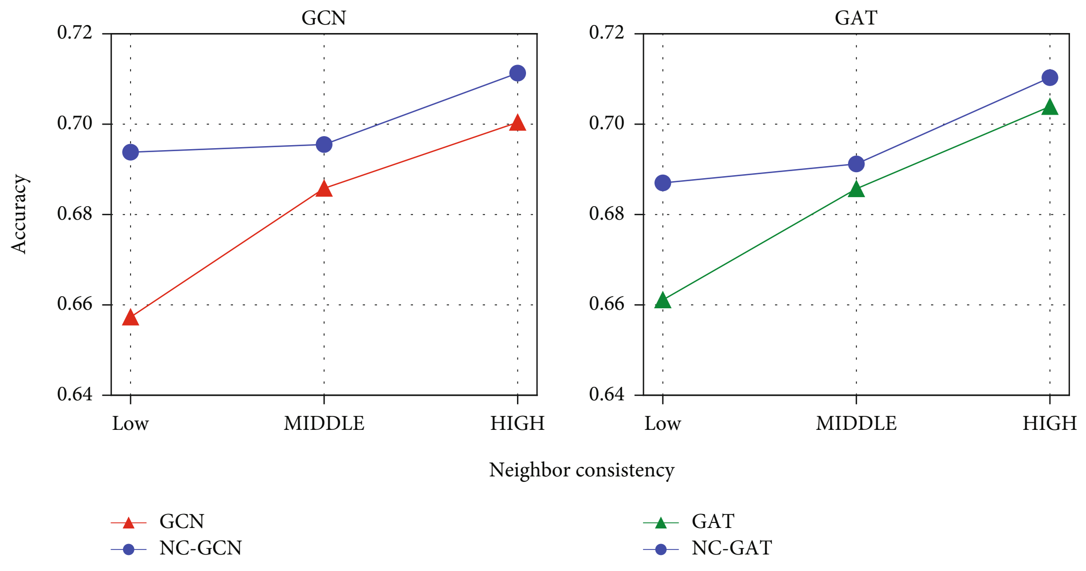
5.4. Node Classification Comparison. To verify the effectiveness of our proposed NC-GNN by reweighting train weights of labeled nodes, we conducted extensive experiments on node classification compared with baselines in benchmark datasets. The results are shown in Table 5. From the table, we can find the following observations:

Our method NC-GNN achieves the best or second-best performance compared with baseline methods in all datasets. The promising results validate the effectiveness of our method which reweighs the labeled nodes with calculated neighbor consistency scores. Compared with NC-GCN, NC-GAT shows fewer improvements compared with corresponding baselines as GAT utilizes an attention mechanism to measure the weights for neighborhoods. The attention scores can alleviate the noise passed from different-labeled neighbors.

DropEdge randomly removes a certain percentage of edges in the graph to improve the generalization performance of graph neural networks, which can be seen as removing different-labeled neighbors randomly. However, the randomicity sometimes discards the important interactions between nodes, resulting in unsatisfying performance.



(a) Cora



(b) Citeseer

Figure 4: The results of classification accuracy in datasets with different neighbor consistency. Low, middle, and high indicate different probabilities of neighbor-inconsistent nodes in the dataset.

Evaluating the neighbor consistency in our method can figure out the different-labeled neighbors without losing the structure information, which is easier to control and stable. SimP-GCN updates the graph structure by combining with kNN-graph calculated by node features' similarity, thus connecting similar nodes and improving homophily in the graph. However, it ignores the different-labeled neighbors which pass unnecessary information in aggregation.

GAT outperforms other baselines as they can weigh neighbor nodes with attention scores so as to prevent the information aggregated from different-labeled neighbors. However, GAT still assumes that all neighbors are beneficial no matter the neighbors' labels. Therefore, the node remains unreliable after aggregating information from different-labeled neighbors. In contrast, our method improves the model by reducing the impact of unreliable nodes in the training procedure. In this way, we can better capture the beneficial information to train the model. The results also show that our method can learn better node embeddings.

Considering the specific dataset, we can find that our method shows more improvements in Citeseer than Cora compared with baseline methods. According to Table 4, we can conclude that we find more neighbor-inconsistent nodes in Citeseer; thus, our method contributes more to reducing the negative impact of different-labeled neighbors and enhancing the performance of the framework. As for PubMed, NC-GNN only finds 6 neighbor-inconsistent nodes, so our method can bring few improvements on baselines.

As for the baseline methods, GCN model performs better than DeepWalk as graph convolution can capture node

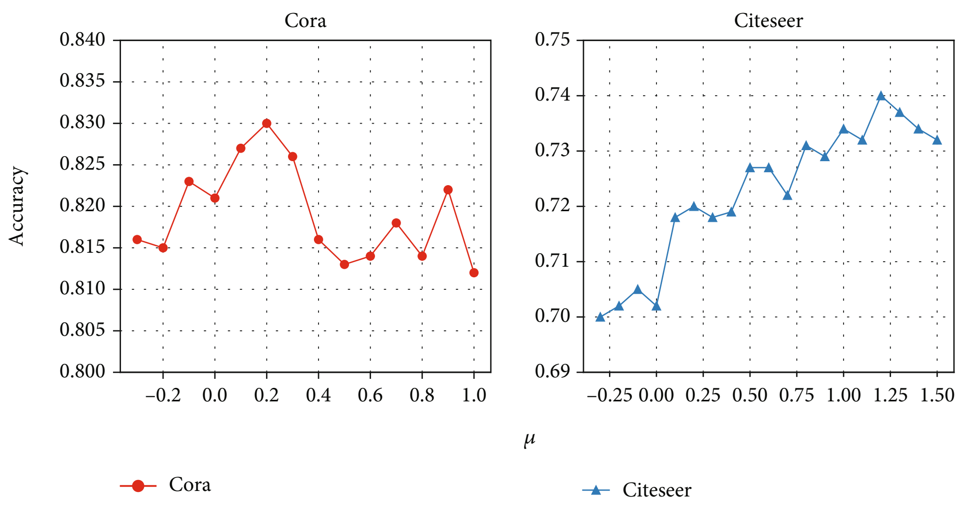


FIGURE 5: Classification accuracy with different μ in Cora and Citeseer.

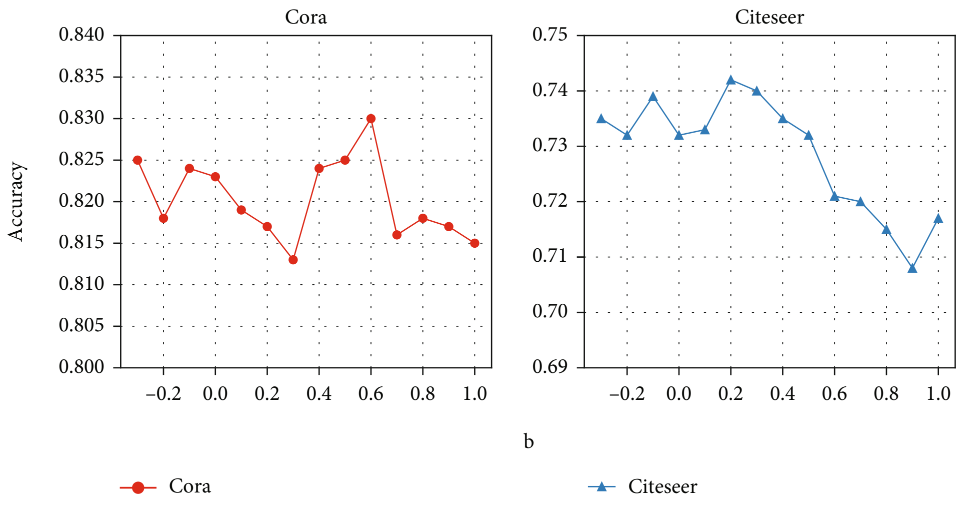


FIGURE 6: Classification accuracy with different b in Cora and Citeseer.

TABLE 6: Running time(s) of the models on the datasets.

Data	Cora	Citeseer	PubMed	Photo	Computers
GCN	1.71	1.87	6.17	3.76	7.75
NC-GCN	2.62(+53.2%)	2.98(+53.0%)	7.04(14.1%)	6.57(+48.1%)	9.58(+23.6%)
GAT	19.50	26.91	89.78	84.14	144.67
NC-GAT	20.39(+4.6%)	28.01(+4.1%)	90.67(+1.0%)	85.95(+2.2%)	146.48(+1.2%)

features and topology information simultaneously. GAT outperforms GCN in some cases as GAT introduces attention to graph convolution to decide the more important neighborhoods. The results are consistent with those in previous works.

5.5. The Influence of Neighbor Consistency. Our method focuses on improving GNN frameworks with the neighbor consistency of labeled nodes. Therefore, the neighbor consistency of the labeled node set plays a considerable influence on the performance of our method. We conduct experiments to discuss the influence of different neighbor consistency probabilities. In particular, we randomly sampled three labeled node sets of 20%, 50%, and 80% neighbor-consistent nodes with the same setting of 20 labeled nodes per class in the previous works. And the probabilities correspond to high, middle, and low neighbor consistency, respectively. We conduct the experiments on Cora and Citeseer with our variants of GCN and GAT. The results are shown in Figure 4.

In Figure 4, we can figure out that as the neighbor consistency grows, all models perform increasingly better, which indicates that neighbor consistency is crucial for model performance. In most situations, our method leads to more improvements on baseline methods when neighbor consistency of labeled nodes is low, while in the high neighbor consistency situation, the gap between our method and baseline decreases as there are fewer neighbor-inconsistent nodes.

5.6. Parameter Study. In this section, we consider the parameters in calculating training weights for labeled nodes in the model, including μ for scaling the node neighbor consistency scores and b for giving the initial training weights. To verify the effects of the corresponding parameters, the experiment results of NC-GCN in Cora and Citeseer datasets with different parameter settings are presented in Figures 5 and 6.

μ tries to scale the computed neighbor consistency scores. A larger μ will highlight the neighbor consistency differences, while a smaller value tries to mitigate the neighbor consistency differences. From Figure 5, we can figure out that NC-GCN performs best in both datasets when the value of μ is positive, which indicates that the labeled nodes with consistent neighbors contribute more in the training process. We can conclude that valuing the neighbor consistency is beneficial for node classification with graph neural networks.

b assigns an initial training weight to the labeled nodes with the corresponding sign given by the indicator method, thus distinguishing between the neighbor-consistent nodes and the neighbor-inconsistent nodes. A positive b indicates that the nodes are vital in the training process, while a negative value weakens the influence. Figure 6 presents that the model achieves the best results in both datasets when b is positive. We can conclude that labeled nodes with consistent neighbors should be more noticed than neighbor-inconsistent nodes in the training process. Besides, the performance of NC-GCN degrades when the value of b is too large, as the model can hardly capture the information of different neighbor consistency scores. So we recommend a small b in model training.

5.7. Running Time Analysis. We report the running time of our method NC-GCN and NC-GAT compared with corresponding baselines in Table 6 (the experiments are conducted in the machine with an Intel(R) Core(TM) i9-10900 K @ 3.70 GHz CPU and a Nvidia GeForce GTX 3090 GPU).

The table shows that our method brings some time cost when the baseline model is simple, whereas there is little running time gain when the baseline is complex, such as GAT. The observation indicates that our method can be applied to massive graphs when choosing a suitable baseline framework. Besides, the time consumption of our method increases less compared with baseline methods when the graph is larger and contains more nodes. This mainly lies in that only partial nodes are labeled in real-world graph tasks.

6. Conclusion

Existing graph neural network methods mostly follow the assumption of homophily, which states that connected nodes are similar and share the same labels. However, the assumption is not always satisfied in real-world graphs. In this paper, we focus on labeled nodes and try to evaluate the consistency between the nodes and corresponding neighborhoods. In particular, we regard information aggregation in graph neural networks as node feature reconstruction and represent the neighborhoods as context features. Then, we design a novel metric neighbor consistency to evaluate the difference between node features and the corresponding context features so as to measure the reliability of labeled nodes after aggregation. Furthermore, we propose the method called *Neighbor Consistent Graph Neural Networks* (NC-GNN) to promote the training of graph neural networks by reweighting the influence of labeled nodes. In this way, the labeled nodes with consistent neighborhoods can contribute more to the model training. Extensive experiments are conducted on benchmark datasets, and outstanding performance indicates the effectiveness of our method.

In the future, we aim to extend the neighbor consistency from labeled nodes to all nodes in the graph to improve aggregation in graph neural networks.

Data Availability

The data used in this paper is available upon the request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This paper is supported by the National Key Research and Development Program of China (Grant No. 2018YFB1403400), the National Natural Science Foundation of China (Grant No. 61876080), the Key Research and Development Program of Jiangsu (Grant No. BE2019105), and the Collaborative Innovation Center of Novel Software Technology and Industrialization at Nanjing University.

References

[1] W. Fan, Y. Ma, Q. Li et al., "Graph neural networks for social recommendation," in *Proceedings of the World Wide Web Conference*, pp. 417–426, San Francisco, California, USA, 2019.

[2] C. Gao, X. Wang, X. He, and Y. Li, "Graph neural networks for recommender system," in *Proceedings of the ACM International Conference on Web Search and Data Mining*, pp. 1623–1625, Virtual Event / Tempe, Arizona, USA, 2022.

[3] W. Jiang, "Graph-based deep learning for communication networks: a survey," *Computer Communications*, vol. 185, pp. 40–54, 2022.

[4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the International Conference on Learning Representations*, Virtual, OpenReview.net, 2017.

[5] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, Virtual, OpenReview.net, 2018.

[6] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 5165–5175, Montreal, Canada, 2018.

[7] M. Zhang, P. Li, Y. Xia, K. Wang, and L. Jin, "Revisiting graph neural networks for link prediction,," 2020, https://arxiv.org.abs/2010.16103.

[8] H. Gao and S. Ji, "Graph u-nets," in *Proceedings of the International Conference on Machine Learning*, pp. 2083–2092, Long Beach, California, USA, 2019.

[9] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *Proceedings of the International Conference on Learning Representations*, Virtual, OpenReview.net, 2019.

[10] H. Yuan and S. Ji, "Structpool: structured graph pooling via conditional random fields," in *Proceedings of the International Conference on Learning Representations*, Virtual, OpenReview.net, 2020.

[11] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: simplifying and powering graph convolution network for recommendation," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 639–648, Xi'an, China, 2020.

[12] J. Sun, Y. Zhang, W. Guo et al., "Neighbor interaction aware graph convolution networks for recommendation," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1289–1298, Xi'an, China, 2020.

[13] A. Benamira, B. Devillers, E. Lesot, A. K. Ray, M. Saadi, and F. D. Malliaros, "Semi-supervised learning and graph neural networks for fake news detection," in *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 568-569, Marriott Downtown, Vancouver, Canada, 2019.

[14] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 7370–7377, Honolulu, Hawaii, USA, 2019.

[15] S. Arora, "A survey on graph neural networks for knowledge graph completion," 2020, https://arxiv.org/abs/2007.12374.

[16] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the International Conference on Machine Learning*, pp. 1263–1272, Sydney, Australia, 2017.

[17] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 1025–1035, Long Beach, California, USA, 2017.

[18] L. Franceschi, M. Niepert, M. Pontil, and X. He, "Learning discrete structures for graph neural networks," in *Proceedings of the International Conference on Machine Learning*, pp. 1972–1982, Long Beach, California, USA, 2019.

[19] C. Zheng, B. Zong, W. Cheng et al., "Robust graph representation learning via neural sparsification," in *Proceedings of the International Conference on Machine Learning*, pp. 11458–11468, Virtual, 2020.

[20] J. Wu, X. Wang, F. Feng et al., "Self-supervised graph learning for recommendation," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 726–735, Virtual, 2021.

[21] L. Xia, C. Huang, Y. Xu, J. Zhao, D. Yin, and J. X. Huang, "Hypergraph contrastive collaborative filtering," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, Madrid, Spain, 2022.

[22] J. Yu, H. Yin, X. Xia, T. Chen, L. Cui, and N. Q. V. Hung, "Are graph augmentations necessary? Simple graph contrastive learning for recommendation," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, Madrid, Spain, 2022.

[23] J. Zhang, M. Gao, J. Yu, L. Guo, J. Li, and H. Yin, "Double-scale self-supervised hypergraph learning for group recommendation," in *Proceedings of the ACM International Conference on Information & Knowledge Management*, pp. 2557–2567, Queensland, Australia, 2021.

[24] W. L. Hamilton, "Graph representation learning, Synthesis Lectures on Artifical Intelligence and Machine," *Learning*, vol. 14, no. 3, pp. 1–159, 2020.

[25] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: a survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 249–270, 2022.

[26] J. Bruna, W. Zaremba, A. D. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," 2014, https://arxiv.org/abs/1312.6203.

[27] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 3837–3845, Barcelona, Spain, 2016.

[28] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Proceedings of the International Conference on Machine Learning*, pp. 6861–6871, Long Beach, California, USA, 2019.

[29] F. Feng, W. Huang, X. Xin, X. He, and T.-S. Chua, "Should graph convolution trust neighbors? A simple causal inference method," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1208–1218, Paris, France, 2021.

[30] H. Yang, X. Yan, X. Dai, Y. Chen, and J. Cheng, "Self-enhanced gnn: improving graph neural networks using model outputs," 2020, https://arxiv.org/abs/2002.07518.

[31] C. Liu, J. Wu, W. Liu, and W. Hu, "Enhancing graph neural networks by a high-quality aggregation of beneficial information," *Neural Networks*, vol. 142, pp. 20–33, 2021.

[32] Y. Zhang, S. Pal, M. Coates, and D. Ustebay, "Bayesian graph convolutional neural networks for semi-supervised classification," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 5829–5836, Honolulu, Hawaii, USA, 2019.

[33] W. Jin, T. Derr, Y. Wang, Y. Ma, Z. Liu, and J. Tang, "Node similarity preserving graph convolutional networks," in *Proceedings of the ACM International Conference on Web Search and Data Mining*, pp. 148–156, Virtual, 2021.

[34] Y. Chen, L. Wu, and M. Zaki, "Iterative deep graph learning for graph neural networks: better and robust node embeddings," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19314–19326, 2020.

[35] X. Chen, Y. Zhang, I. Tsang, and Y. Pan, "Learning robust node representations on graphs," 2020, https://arxiv.org/abs/2008.11416.

[36] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5812–5823, 2020.

[37] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, “Graph contrastive learning with adaptive augmentation,” in *Proceedings of the Web Conference*, pp. 2069–2080, Virtual, 2021.

[38] M. Luo, X. Chang, L. Nie, Y. Yang, A. G. Hauptmann, and Q. Zheng, “An adaptive semisupervised feature analysis for video semantic recognition,” *IEEE Transactions on Cybernetics*, vol. 48, no. 2, pp. 648–660, 2018.

[39] D. Zhang, L. Yao, K. Chen, S. Wang, X. Chang, and Y. Liu, “Making sense of spatio-temporal preserving representations for EEG-based human intention recognition,” *IEEE Transactions on Cybernetics*, vol. 50, no. 7, pp. 3033–3044, 2020.

[40] K. Chen, L. Yao, D. Zhang, X. Wang, X. Chang, and F. Nie, “A semisupervised recurrent convolutional attention model for human activity recognition,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 5, pp. 1747–1756, 2020.

[41] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, “Pitfalls of graph neural network evaluation,” 2018, https://arxiv.org/abs/1811.05868.

[42] M. Wang, D. Zheng, Z. Ye et al., “Deep graph library: a graph-centric, highly-performant package for graph neural networks,” 2019, https://arxiv.org/abs/1909.01315.

[43] M. Fey and J. E. Lenssen, “Fast graph representation learning with PyTorch Geometric,” *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[44] B. Perozzi, R. Al-Rfou, and S. Skiena, “Deepwalk: online learning of social representations,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701–710, New York, New York, USA, 2014.

[45] Y. Rong, W. Huang, T. Xu, and J. Huang, “DropEdge: towards deep graph convolutional networks on node classification,” in *Proceedings of the International Conference on Learning Representations*, Virtual, OpenReview.net, 2020.

Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 9905114, 13 pages
https://doi.org/10.1155/2022/9905114

Research Article

Mining and Application of Tourism Online Review Text Based on Natural Language Processing and Text Classification Technology

Hongsheng Xu[1,2] **and Yanqing Lv**[1,2]

[1]*College of Electronic Commerce, Luoyang Normal University, Luoyang, 471934 Henan, China*
[2]*Henan Key Laboratory for Big Data Processing & Analytics of Electronic Commerce, Luoyang Normal University, Luoyang, 471934 Henan, China*

Correspondence should be addressed to Hongsheng Xu; xhsls@lynu.edu.cn

Received 1 April 2022; Revised 2 May 2022; Accepted 8 May 2022; Published 23 May 2022

Academic Editor: Lei Zhang

This paper firstly describes the research status of online review text mining and finds out the problems existing in the mining and application of tourism texts. Aiming at these problems, this paper proposes a text mining method for tourism online reviews based on natural language processing and text classification technology. The first step is to analyze the validity of the online review text; the purpose is to remove the invalid text and improve the mining efficiency of the online review text. The second step is to conduct a comprehensive evaluation of scenic spots and hotels based on text classification technology and sentiment analysis. The comprehensive evaluation indicators are established for the five core service contents. High-quality scenic spots and hotels are selected according to the ranking of comprehensive evaluation. The third step is to propose a mining method of tourism hot words based on natural language processing for the selected high-quality tourist locations. The obtained hot words can intuitively show the impression of tourists on the scenic spot. The fourth step is to use mutual information combined with the left and right entropy to discover new words and to mine service characteristics of high-quality scenic spots and hotel from the new words. Finally, the proposed new methods are tested on the crawled tourism online review texts. The experimental results show that the novel comprehensive evaluation method proposed in this paper can truly and objectively select high-quality scenic spots and hotels and provide an important basis for the decision-making of tourism management. On this basis, hot words and new words can be effectively excavated from relevant online review texts, and travel impressions can be fed back from various aspects and angles.

1. Introduction

With the booming development of tourism, people pay more and more attention to practical experience when traveling. Relevant government departments and tourism enterprises are also focusing on improving the service quality of the tourism industry. Major travel websites, such as Trip, TravelGo, and eLong, all provide a wealth of tourist comment functions. Tourists can comment on scenic spots or hotels in tourist destinations from various aspects. These online review texts generally reflect the visitor's experience. These review texts can not only provide reference for other tourists but also provide suggestions for improving and enhancing tourism services for the operation department of the scenic spot, the local cultural tourism management department, and tourism enterprises [1].

Online review texts are often created by thousands or even tens of thousands of tourists and are a type of user generated content (UGC). In recent years, with the help of modern computers and artificial intelligence, especially natural language processing technology, these text data can be automatically processed and analyzed to obtain indicators reflecting the impression of tourists, thus providing a decision-making reference for the development of tourism [2]. At present, many scholars have carried out research on improving tourism services by analyzing online review texts.

This article uses the blog post about Zhujiajiao in Sina blog to divide the tourism perception image of Zhujiajiao from the direction of text mining [3]; Cai et al. [4] studied the audience perception of urban tourism image in Guiyang City based on ROST text mining software. This article used text mining method to conduct word frequency, sentiment, and semantic network analysis of online travel notes about Gansu tourist spots based on four typical travel websites such as Trip, Mafengwo, Lvmama, and Tuniu [5]. Li et al. [6] used text mining to conduct image perception research on the comments of tourists on typical urban tourism communities in Beijing, such as Baidu Travel, https://Trip.com/. This article conducted a comparative study of tourists' opinions and suggestions based on text mining in Guangxi Qinbeifang. This paper also formulated tourism policies for the government and related tourism management departments and provided an important direction for the development of tourism in Guangxi Qinbeifang and has important practical significance [7].

After analyzing the current research status, it is found that the above research is not systematic in the application of online review texts, lacks a comprehensive evaluation mechanism for scenic spots and hotels, and cannot reflect tourists' travel impressions from multiple angles and levels, so it cannot effectively feedback the services and characteristics of hotels and scenic spots. This paper proposes the analysis and processing of online review text based on natural language processing and text classification technology. The first is the validity analysis of online review text; the purpose is to remove invalid text. The second step is to use the text classification technology combined with the sentiment analysis method to carry out the comprehensive evaluation of the scenic spots and hotels, to comprehensively score the five services that the tourists are concerned about, and to select the high-quality tourist spots and hotels. The third step is to use the named entity recognition method in natural language processing to mine hot words for the top-ranked scenic spots and hotels. The obtained hot words can effectively feedback tourists' intuitive impression of scenic spots and hotels. The fourth step is to analyze the characteristic services of scenic spots and hotels based on new word discovery. The service characteristics of high-quality scenic spots and hotels are excavated in the new words. Therefore, these methods can reflect tourists' travel impressions from multiple perspectives.

2. Related Technical Analysis

2.1. Text Classification Techniques

2.1.1. Bayesian Methods and Naive Bayes. Bayesian methods and theories were first proposed by British mathematician Thomas Bayes. In recent years, with the development of artificial intelligence, especially the rise of machine learning, data mining, and other technologies, Bayesian theory has a broader development and application space.

The Bayesian classifier is a general term for a class of classification algorithms, which are all based on Bayes' theorem. Their applications in text mining are mainly focused on plain Bayesian classifiers and Bayesian network classifiers. The Naive Bayes is the simplest and most common type of Bayesian classifier. The algorithm assumes that the probability of all words appearing in the text is considered to be relatively independent [8].

Assuming that the set X is the set of text categories, determining whether a text y belongs to a category x_i can be done by calculating the probability of $P(x_i \mid y)$, i.e., given a text y, calculate what is the probability that it belongs to the text category x_i. The discriminant rule of plain Bayes is to categorize y into the category that makes $P(x_i \mid y)$ reach the maximum probability, i.e., to solve $\text{argmax}P(x_i \mid y)$.

The Naive Bayes is the simplest and the most common of the Bayesian classifiers. The algorithm assumes that the likelihood of each attribute taking values is considered to be independent and uncorrelated with the values of other attributes [9]. The core idea is that for a given item to be classified, the probability of occurrence of each category under the conditions of this item is solved, and whichever is the largest is considered to be the category to which this item to be classified belongs.

Let X be an unlabeled data sample, and let H be some assumption that the data sample X belongs to a particular class C. In the classification problem, we want to obtain $P(H \mid X)$, i.e., given the predicted data sample d, the probability that the assumption H holds, and the classification is completed by comparing the maximum probability [10]. Its Bayes' theorem is formulated as follows:

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}, \tag{1}$$

where $P(H \mid X)$ is the posterior probability, which denotes the probability of occurrence of H given that condition X is found. $P(H)$ is the prior probability, which denotes the prior probability of hypothesis H. $P(X)$ is the prior probability of condition X. $P(H)$ is independent of X.

The process of Naive Bayes is as follows.

Each data sample is represented by an n-dimensional feature vector $d = \{x_1, x_2, x_3, \cdots, x_n\}$, which describe the n attributes $d_1, d_2, \cdots, d_n$ samples with n metrics. Assume m classes $c_1, c_2, \cdots, c_n$. Given an unknown data sample d, the classification method will predict that d belongs to the class with the highest posterior probability. That is, the unknown data sample is assigned to class c_i.

The class c_i whose $P(c_i \mid d)$ is the largest is called the maximum a posterior hypothesis and according to Bayes' theorem.

$$P(c_i|d) = \frac{P(d|c_i)P(c_i)}{P(d)}. \tag{2}$$

Since $P(d)$ is constant for all classes, it is only necessary that $P(d \mid c_i)P(c_i)$ is maximal. After calculating $P(d \mid c_i)P(c_i)$ for each class, the sample d is assigned to the class c_i whose $P(d \mid c_i)P(c_i)$ is the largest.

For the calculation of the probability estimate, to improve its accuracy, the Laplace smoothing estimate can be used with the following equation.

$$P(c_j) = \frac{\sum_{i=1}^{|D|} P(c_j|d_i)}{|D|} (j = 1, 2, \cdots, |C|), \tag{3}$$

$$P(d_i|c_j) = \frac{1 + \sum_{t=1}^{|D|} B_{it} P(c_j|d_i)}{2 + \sum_{t=1}^{|D|} P(c_j|d_i)} (j = 1, 2, \cdots, |C|; t = 1, 2, \cdots, n), \tag{4}$$

where D is the training text set and $P(c_j \mid d_i) \in \{0, 1\}$, indicating whether the training text d_i belongs to the text of class c_j, which "1" means belonging, and "0" means not belonging.

2.1.2. Classifier of Linear Support Vector Machine. Support vector machine (SVM) algorithm is considered to be one of the more effective methods in text classification, which is a machine learning method based on statistical learning theory [11]. This technique solves the previous problem of requiring an infinite number of samples by simply abstracting a certain amount of text into vectorized training text data through computation, which improves the accuracy of the classification and can be widely used in statistical classification and regression analysis. It maps the vectors into a higher dimensional space in which a maximum interval hyperplane is established [12]. Two hyperplanes parallel to each other are built on either side of the hyperplane separating the data, and the separation hyperplane maximizes the distance between the two parallel hyperplanes. It is assumed that the greater the distance or gap between the parallel hyperplanes, the smaller the total error of the classifier.

Definition 1. For a given data set that is linearly divisible, this equivalent method of using interval maximization or solving the corresponding convex quadratic programming problem. The separation hyperplane is learned as $\omega^* x + b^* = 0$, and the corresponding classification decision function is $f(x) =$ sign $(\omega^* x + b^*)$. It is called linearly separable SVM.

Definition 2 (geometric interval). For a given training sample data set M and hyperplane (ω, b), we call the function interval of the hyperplane about the sample point (x_i, y_i) as $\gamma_i = y_i(\omega/\|\omega\| \cdot x_i + b/\|\omega\|)$. The minimum value of the geometric interval between the hyperplane (ω, b) and all sample points in the data set M is $\gamma = \min_{i=1,2,\cdots n} \gamma_i$.

SVM has few parameters, and the most important one is the kernel function. When there are relatively many text features, the linear kernel function is enough. The SVM using linear kernel function is called linear SVM; it is also the kernel function used in this paper.

SVM (support vector machine) is a statistical theory classification algorithm. The algorithm implements sample features to find a balance between model complexity and learning ability. It is more effective in solving small samples, nonlinearity, and high dimensionality.

The optimal classification plane $\omega \cdot \phi(x) + b = 0$ exists for a sample S. The original parameter space is transformed to a high-dimensional space using a nonlinear function, and then, the above hyperplane is established; ω denotes the plane normal vector, and b denotes the intercept set to separate the plane function:

$$\omega \cdot \phi(x) + b = 0. \tag{5}$$

The discriminant function $h(x) = \omega \cdot \phi(x) + b$; $h(x)$ will be normalized, so that the samples meet $h(x) \gg 1$; the operation transformation can be obtained after the simplification of the formula $D = 2/\|\omega\|$, so that the classification interval is the maximum is equivalent to make $\|\omega\|$ minimum.

To satisfy the above conditions, the following conditions are needed to make the classification models work correctly for all samples [13].

$$y_i(\omega \cdot \varphi(x_i) + b) - 1 \gg 0 (i = 1, 2, \cdots, n). \tag{6}$$

For each inequality constraint introduce a Lagrange multiplier (Lagrange multiplier) $\alpha_i \geq 0, N\alpha_i \geq 0, i = 1, 2, \cdots, N$; construct the Lagrange function:

$$L(\omega, b, a) = \frac{\|\omega\|^2}{2} - \sum_{i=1}^{n} \alpha_i \left(y_i \left(\omega^T \varphi(x_i) + b \right) - 1 \right). \tag{7}$$

By eliminating ω and b, the original constrained optimization problem can be equated to the minimax dual problem.

When linearly inseparable, it needs to be transformed to a higher dimensional space to make it linearly separable. In this case, the relaxation variable method is needed to solve this kind of problem, and a relaxation variable $\delta \geq 0$ is introduced for each sample so that the constraint becomes:

$$y_i \left(\omega^T \varphi(x_i) + b \right) \geq 1 - \delta_i. \tag{8}$$

The objective function then becomes:

$$\frac{\min \|\omega\|^2}{2} + C \sum_{i=1}^{i} \delta_i. \tag{9}$$

The larger C is the penalty factor, the smaller the number of error points, but it should not be too large to avoid excessive clutching. To avoid dimensional catastrophe, it is necessary to introduce the kernel function, which is a mapping from low-dimensional space to high-dimensional space.

2.2. Mutual Information and Left-Right Entropy Theory

2.2.1. Information Entropy. Information entropy was originally proposed by Shannon and can be used to describe the uncertainty of an event. Usually, a text with a lower probability of occurrence of an event contains more information and thus has a higher information entropy [14]. For example, "Bill Gates goes bankrupt" has a much lower probability of occurring than "Bill Gates becomes the richest man," so the former has a higher information entropy [15]. The formula of information entropy can be expressed as follows:

$$H = -\sum_{i=1}^{n} P_i \cdot \log_2(P_i). \tag{10}$$

2.2.2. Information Gain. Information gain (IG) (Mitchell1 1997) represents the average information of a document class when a text contains a certain feature, defined as the difference in information entropy before and after a feature appears in the text [16]. Assume that C is the set of text classes, c is the text class variable, d is the text, and t is the feature. For feature t, its information gain is denoted as $\mathrm{IG}(t)$. The frequency of documents with and without t occurring in c is examined. It is used to measure the information gain of word t for category c. The calculation formula is shown as follows:

$$\mathrm{IG}(t) = P(t)\sum_{i=1}^{|C|} P(c_i|t)\log\frac{P(c_i|t)}{P(c_i)} + P(\bar{t})\sum_{i=1}^{|C|} P(c_i|\bar{t})\log\frac{P(c_i|\bar{t})}{P(c_i)}. \tag{11}$$

2.2.3. Mutual Information. Mutual information refers to the amount of information contained in one random variable with another random variable and also a measure of the association between two variables. The formula of mutual information can be expressed as follows [17].

$$I(X, Y) = H(Y) - H(Y|X) = H(X) - H(X|Y) = H(X) + H(Y) - H(X, Y). \tag{12}$$

Mutual information (MI) is a commonly used information metric in information theory and is widely used in statistical language models. It measures the importance of a word to a category based on the occurrence of the word t. The calculation formula is shown as follows:

$$\mathrm{MI}(t) = \sum_{i=1}^{|C|} P(c_i)\log\frac{P(t|c_i)}{P(t)}, \tag{13}$$

where $P(t|c_i)$ denotes the conditional probability that feature t appears in category c_i, $P(c_i)$ denotes the probability that the text with category c_i appears in the document collection, and $P(t)$ is used to denote the probability that features t appears in the text.

From the formula, it can be seen that feature selection using mutual information prefers to select low-frequency feature words. That is, if two feature words have the same conditional probability $P(t|c_i)$, the feature word with fewer occurrences will receive a higher mutual information value than the feature word with more occurrences.

3. Validity Analysis of Online Review Texts

Online reviews sometimes have irrelevant content, simple copying and modification, and irrelevant content, which prevents tourists from obtaining valuable information from online reviews and also brings challenges to the operation of various online platforms [18, 19]. Therefore, it is very important to analyze the validity of online review texts. Validity analysis can be regarded as a two-classification problem of text, and it is divided into two categories: valid and invalid. That is, the subjectivity, randomness, irrelevant content, and suspicious copy and paste text in the sentence are classified into the "invalid" category, and the rest of the comment text belongs to the "valid" category [20]. The sample tested in this paper uses the web crawler technology Python 3.6 to crawl the review texts of scenic spots and hotels in China's major travel websites, especially choosing the online travel giant Trip as the main source of sample data.

The main steps about the validity analysis of online review texts are described below.

3.1. Data Preprocessing. To analyze the validity of the review content, it is necessary to preprocess the review content of scenic spots and hotels. The following two steps are included:

Step 1. Filter punctuation marks and stop words. Because there are many useless symbols in text information, many words have no practical meaning, such as some words. Therefore, the review data must be filtered first. The processing steps include filtering punctuation marks, special symbols, removing stop words, useless adverbs, etc. Here, we use a custom stop word dictionary to remove stop words more accurately and efficiently and to filter out a large number of words such as adverbs that have no practical meaning by word filtering.

Step 2. Word segmentation processing. Chinese word segmentation is not as simple as English word segmentation. There is no obvious distinguishing mark between words, and semantic and logical relationships are often taken into account. The effect of word segmentation directly affects information analysis and experimental results. At present, several common word segmentation tools are Jieba word segmentation, SnowNLP word segmentation tool, and HanLP word segmentation tool.

Jieba word segmentation is the most widely used word segmentation technology in China. The Jieba word segmentation has the following features: precise mode can segment sentences most accurately and is suitable for text analysis; the full mode can scan all the words that can be turned into words in the sentence, and the speed is very fast. SnowNLP word segmentation is a Python-based library. Its functions are relatively simple, and it is relatively easy to use. Jieba word segmentation is more suitable for text analysis than SnowNLP. Therefore, this paper uses Jieba word segmentation for Chinese word segmentation and word annotation.

3.2. Manual Annotation Validity Information. To build a machine learning model to analyze the effectiveness of reviews, the model must first be pretrained. So we made 1000 valid manual annotations in the reviews of scenic spots and hotels in advance. If there is useful information in the reviews of hotels and scenic spots, we regard it as a valid mark of "1." For some irrelevant or useless information, we regard it as an invalid mark of "0." The models are trained and tested accordingly through these manually annotated data.

Here, there are nearly 60,000 reviews of scenic spots and more than 25,000 reviews of hotels collected through crawler technology. Take the scenic spot as an example; we extract 1,000 items from them for manual annotation, that is, to add a column to the original data file. The column name is set to "valid," which is set to 1 when valid and 0 when invalid.

3.3. Extract Text Features. For the comment content of the segmented words, the numerical calculation of text feature is performed. In this paper, document frequency is used to calculate the value of text feature. Document frequency (DF) is an efficient feature selection algorithm that counts how many texts contain the word in the entire dataset. Its document frequency is counted for each feature in the training text set, and those features with a particularly low and particularly high document frequency are removed according to a preset threshold. DF is the simplest feature selection method, and this method has low computational complexity and can perform large-scale classification tasks. It is a common method for feature dimension reduction.

3.4. Construction of a Classification Model for the Validity of Online Review Texts Based on Naive Bayes Classifier. The classification method based on Naive Bayes classification algorithm is divided into two stages. The first stage is the training stage, building a classifier with a known set of instances. The known instance set used to build the classifier is called the training instance set, and each instance in the training instance set is used as a training instance. Since the class labels of the training instances are known, the construction of the classifier is a tutored learning process.

The second phase is the test phase, using constructed classifiers to classify unknown instances. The classifier generally needs to be evaluated before it can be used to predict. Only classifiers with the required classification accuracy can be used to classify test instances.

The classifier used here is the Naive Bayes classifier, and its characteristics mainly include incremental learning. Prior knowledge can determine the final probability of the hypothesis together with the observed instances and allow assumptions to make uncertain predictions. The classification of new instances can be predicted by multiple assumptions together weighted by their probabilities. Based on the above classification characteristics, this paper constructs a classification model for the validity of online review texts based on the Naive Bayes classifier. The specific process is using the Naive Bayesian method for supervised learning and training based on the 1,000 manually annotated scenic spot reviews. These trained models are then used to annotate all the online review texts. Then, for the annotated review file table of hotels or scenic spots, a column is added to the right: "validity," and it is marked as "valid" or "invalid" according to the output of the trained model. Finally, the validity analysis table of tourism online review texts is obtained, as shown in Table 1.

4. Comprehensive Evaluation of Scenic Spots and Hotels Based on Text Classification Technology and Sentiment Analysis

After removing the invalid text, the effective online review texts of scenic spots and hotels are obtained here. We use text classification techniques to classify online review texts into appropriate categories. According to the five aspects of scenic spots and hotels that tourists focus on, service, location, facilities, hygiene, and cost, a comprehensive evaluation is carried out, and an evaluation model is constructed according to mean squared error (MSE) combined with sentiment analysis. Mean squared error is a measure that reflects the degree of difference between the estimator and the estimated. Let t be an estimator of the population parameter θ determined from the subsample. The mathematical expectation of $(\theta - t)^2$ is called the mean squared error of the estimator t. $\text{MSE} = \sigma^2 + b^2$, where σ^2 and b^2 are the variance and bias of t, respectively.

Therefore, this paper uses the mean squared error to build the evaluation model according to the above data analysis. The specific steps are as follows.

(1) This paper uses web crawler technology to crawl corpus in five aspects of service, location, facility, hygiene, and cost from major tourism websites and performs part of manual annotation and model training

First, we look for the text classification corpus for training, where the text files containing about 5,000 hotel online reviews are collected. According to the five aspects of service, location, facilities, hygiene, and cost, manual annotation is carried out one by one, and more than 1,500 online comment texts are marked and classified into the abovementioned five categories.

(2) The paper splits the review texts of hotels or scenic spots into single sentences and then classifies the texts according to service, location, facilities, hygiene, and cost

According to the data characteristics of online review texts in the tourism field, two suitable text classification methods are selected for comparative testing. The best text classification method is selected according to the processing speed and classification evaluation indicators.

Two text classification methods are used here: one is Naive Bayesian method, and the other is linear SVM. The sample data are tested separately, and the experimental

Table 1: Validity analysis table of tourism online review texts.

Hotel name	Comments	Comment validity
H01	The hotel is suitable for family travel.	1
H01	Upgraded the room, late checkout, and it is great.	1
H01	I have come to Guangzhou every year, and I will stay in a ** hotel, because the location is good and the price/performance ratio of the hotel is also good.	1
H02	The hotel is very good.	0
H02	Super 5 stars.	0
H02	The hotel is clean and hygienic; the service is very good!	1
H03	The hotel is a traditional standard five-star hotel. The only regret is that the bathroom only has a bathtub and no shower.	1
H03	Hotel facilities are a bit old, but the location is really good.	1
H03	Very good location and convenient travel.	1
H04	Good location, clean, and tidy room.	1
H04	The hotel is in a great location! 100 m out of the metro entrance. And the service is fantastic! The hotel management is improving!	1
H04	The hotel is a good choice.	0
H05	It is very close to Guangzhou East Railway Station, convenient for picking up and dropping off guests, the price is reasonable and acceptable, the breakfast is good and filling, and the variety is complete, suitable for young and old women and children to choose their own food.	1
H05	Jianguo is always a very good choice for business travelers who want to sleep in a serviced area in the Central district of Tianhe.	0
H05	High cost performance, rich breakfast varieties, many kinds of bread, close to the subway.	1
H06	It is very close to Guangzhou East Railway Station and subway station, and it is very convenient to eat nearby. The price/performance ratio is quite good. Recommended to stay.	1
H06	The room is very good; the service is very good; the front desk is very good.	1
H06	Very good for many times and cost-effective.	1
H07	The hotel is relatively close to Guangzhou East Railway Station, just next to it.	1
H07	Hygiene is very good.	1
H07	Good location and good service.	1
H07	The hotel is very good; the service is very good.	1

results show that the linear SVM method is faster in terms of data processing speed, as shown in Figure 1.

In this paper, the indicator test of the text classification model is carried out. The model evaluation indicators of the classification algorithm are often measured by the confusion matrix, as shown in Table 2. The four data obtained from the confusion matrix are extended by calculation to obtain four secondary metrics: accuracy, precision, recall, and F1-score, which are the core metrics for evaluating the classification model [21].

Accuracy of a classification model (accuracy) represents the ratio of samples correctly predicted by the model to all samples, and in general, the higher the accuracy, the better the classifier is accordingly. This is shown in the following equation.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}. \tag{14}$$

The precision of the classification model (precision) is defined as the percentage of samples with true positive class among all samples predicted to be positive class, and the formula is as follows:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{15}$$

The recall (recall) of a classification model is defined as the percentage of samples with true positive classes that are correctly predicted, and the formula is as follows:

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{16}$$

F1-score is the summed mean of precision and recall, see equation (17), which combines the results of precision and recall and is closer to the smaller of the two, so when precision and recall are close, F1 is the largest. A higher value of F1 indicates a better model prediction.

$$\text{F1} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}. \tag{17}$$

Here, the test is carried out on the sample data. First, the

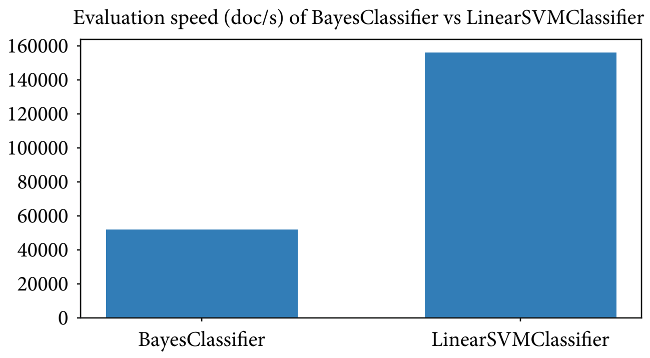


Figure 1: Comparison of the processing data speed between the two classification methods.

Table 2: Confusion matrix.

Predict True	0	1
0	TP	FN
1	FP	TN

Naive Bayes method is used. According to the above four classification indicators, the test results in five aspects of service, location, facilities, hygiene, and cost are shown in Figure 2.

In order to compare the classification effect, this paper uses the linear support vector machine to test the five aspects (service, location, facilities, hygiene, cost) on four classification indicators. The test results are shown in Figure 3. The experimental results show that for the online review texts tested in this paper, the linear support vector machine has better classification effect than Naive Bayes method on the four classification indicators by synthesizing the five aspects of tourists' concerns. Finally, linear support vector machine is selected as the classification model of online review text.

(3) A single evaluation is made on the 5 aspects of each hotel or scenic spot (service, location, facilities, hygiene, cost); combined with text sentiment analysis, this paper adopts a 5-point system for comprehensive scoring

On the basis of text classification, all online review texts of hotels can be classified into the above 5 categories (service, location, facilities, hygiene, cost). After the classification is done using the classifier, a confusion matrix with specific numerical values can be shown by Figure 4.

For example, in the "service" category, a certain online review under this category is scored according to objective criteria. Sentiment analysis in natural language processing is used here. Sentiment analysis is a classification technique based on natural language processing. The main purpose of sentiment analysis methods is to determine whether a review is positive or negative. Therefore, this paper uses sentiment score indicators to quantify online review texts. Sentiment analysis generally sets the sentiment of the text to a value between (0,1). 0.5 represents neutrality, a score closer to 0 represents a negative emotion, and a score closer to 1 represents a positive emotion. Since the score is based on a 5-point system, the result of sentiment analysis is multiplied by 5 to get a score between (0,5). After all the online review texts of the hotel under the category of "service" are scored on the above 5-point system and averaged, the "service" score of the hotel can be obtained.

In the same way, the other four aspects of the hotel can be scored. In addition, the same method can be used to score the five aspects of the scenic spot.

(4) The evaluations of all hotels or scenic spots are normalized and comprehensively evaluated, and a score between 1 and 5 is obtained. The experimental results are shown in Figure 5

The experimental results in Figure 5 show that Hotel01's scores in five aspects are 4.5, 4.6, 4.3, 4.5, and 4.7, and the comprehensive score is 4.5; Hotel03's scores in five aspects are 4.3, 4.6, 4.1, 4.5, and 4.4, and the comprehensive score is 4.4; Hotel05's scores in five aspects are 4.4, 4.6, 3.9, 4.5, and 4.5, and the comprehensive score is 4.4. Both Hotel02 and Hotel04 have the same comprehensive score of 4.2. Therefore, this paper uses the sentiment analysis method in natural language processing to comprehensively score the five services of the hotel, and the hotel with the highest score can be selected as Hotel01, which can provide decision-making basis for hotels and scenic spots to improve service quality.

5. Mining and Analysis of Tourism Hot Words Based on Natural Language Processing

Hot words are the most direct form to reflect tourists' impression of scenic spots and hotels. Generally speaking, whether it is positive or negative, tourists will always have some representative words in their comments on scenic spots and hotels. These words are often also high-frequency words, which are obtained by statistical methods after excluding irrelevant stop words. The hot words need to be obtained through natural language processing. This paper obtains the hot words in the online review text based on the named entity recognition method.

Named entity recognition is a common basic task in natural language processing, and it is also an important component of tasks such as information extraction, question answering, syntactic analysis, and machine translation. Common types of entities are person names, place names, institution names, times, dates, money, and so on. Different tools have some subtle differences in the distinction between entity types. This article is concerned with a few types of place names, such as hotels or scenic spots [22].

In this paper, the idea of mining hot words from online review texts is as follows. Firstly, according to the above method, the scenic spots and hotels are comprehensively evaluated and scored, and the thresholds are set into three levels: high, medium, and low, and the top 10 of

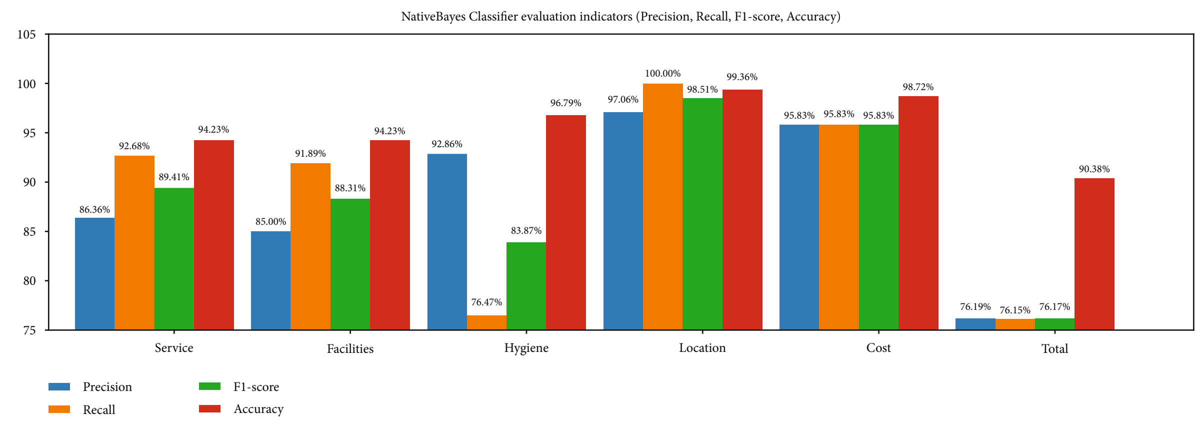


Figure 2: Results of the classification indicator test based on Naive Bayes.

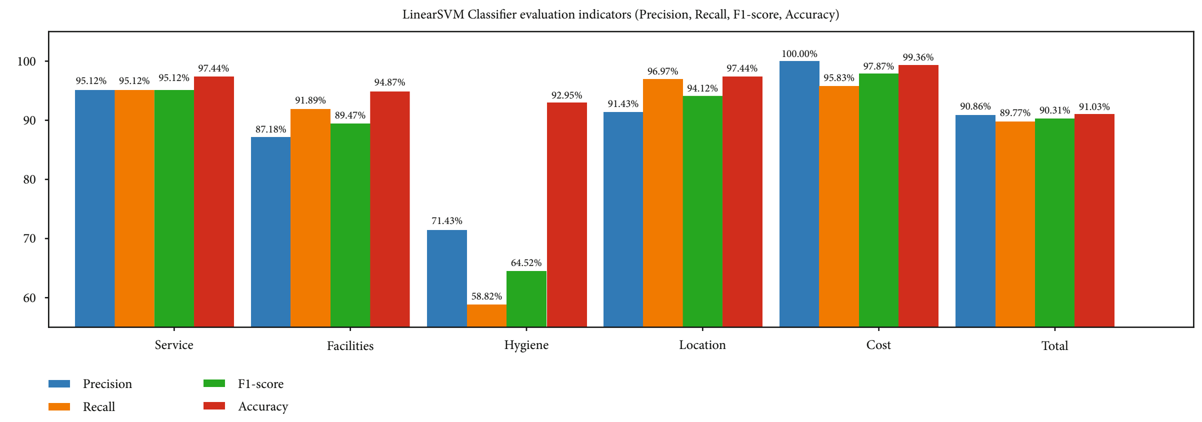


Figure 3: Results of the classification indicator test based on linear support vector machine.

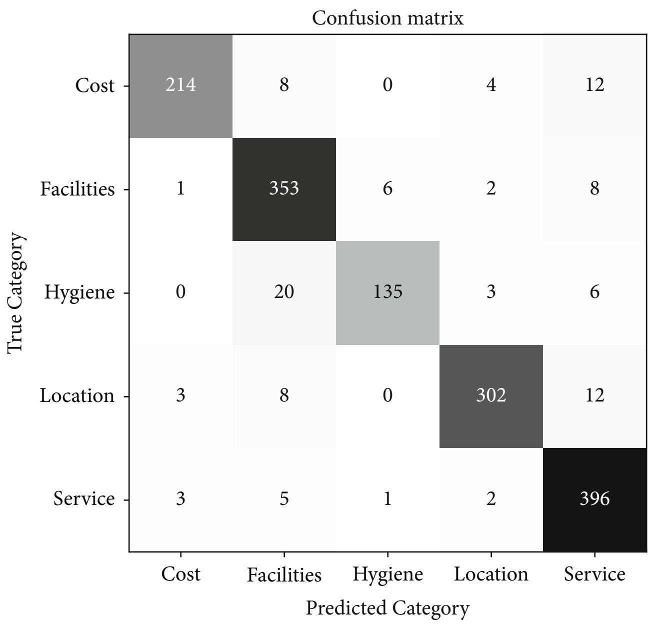


Figure 4: Confusion matrix with specific values.

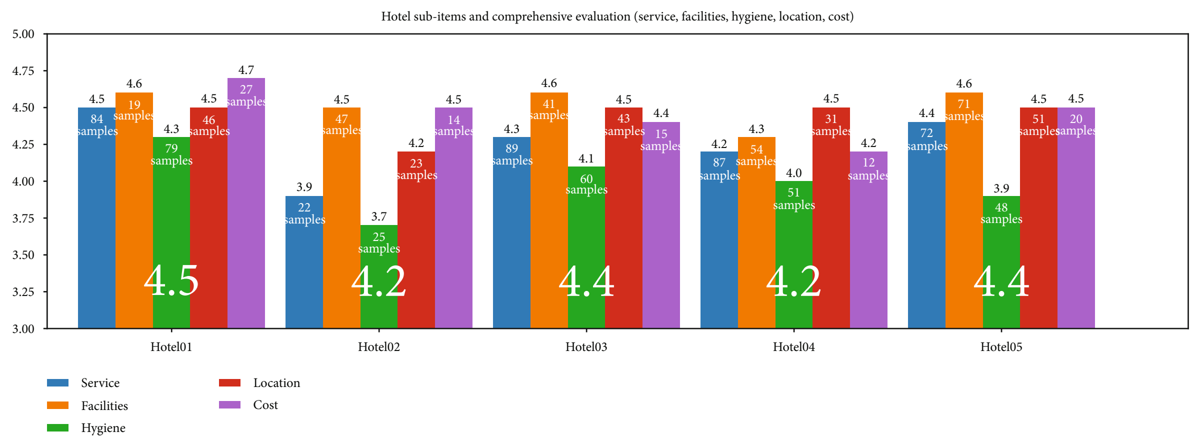


Figure 5: Results of comprehensive hotel scoring based on natural language processing and sentiment analysis.

each level are taken as the online comment text for mining hot words. These online comment texts are then named entity recognition, and the obtained named entities (noun) are stored in the word segmentation dictionary. Then, the comment text is segmented based on the named entity as a word segmentation dictionary, and after filtering the part of speech, high-frequency words are counted, and finally, hot words are obtained. Based on this idea, a flow chart of hot word acquisition is formed, as shown in Figure 6.

(1) *Corpus Preprocessing*. Here, the Python module of pandas is used to read the corpus according to the scenic spots, and the Python module of the natural language processing tool SnowNLP is used to segment superlong text into short sentences. Since the NLP tool (HanLP 2.x) used in the following steps cannot handle superlong text, during preprocessing, these superlong sentences need to be segmented into short sentences to lay the foundation for subsequent processing

(2) *Named Entity Recognition*. Using the NER function in the NLP tool, the words with named entities such as scenic spots and facilities are extracted from each sentence and written to the word segmentation dictionary

After comprehensive testing and comparison, the NER function provided in HanLP 2.x version is adopted here. The reason is that this version uses a large-scale Chinese corpus pretrained deep learning model and provides three mature methods: ner/msra, ner/pku, and ner/ontonotes. The effect of the test here is relatively good, and all the scenic spots can be found. Here, the named entities obtained by the NER methods are merged and deduplicated as the actual NER results.

(3) *Word Segmentation*. In the case of using a word segmentation dictionary of NER results, the word segmentation tool is Jieba. This paper uses this tool to perform word segmentation on all the review corpus of a scenic spot or hotel. In order to ensure the effect, only the noun is retained in the result after word segmentation

(4) *Statistics of Word Frequency*. For the list of word segmentation results, the word frequency of all words is counted. The 20 words with the highest frequency are found and written to the file in the required format

The hot words generated by 16 scenic spots such as A01-A16 are listed in Table 3. From the experimental results in Table 3, it can be seen that hot words can effectively display the intuitive impression of scenic spots to tourists.

6. Analysis Characteristic Services of Scenic Spots and Hotels Based on New Word Discovery

In order to attract tourists and enhance their competitive advantage, scenic spots need to find their own characteristics. This paper excavates their respective characteristics and highlights from the online review texts of scenic spots and hotels. Combined with the comprehensive evaluation results of scenic spots and hotels, after the appropriate threshold value is determined, the scenic spots and hotels are divided into three levels: high, medium, and low. And for the top three of each level, this paper applies the new word discovery method to find the characteristics of scenic spots and hotels.

New word discovery is also called “new word recognition” or “unregistered word recognition.” The new words here are words that are newly generated with the development of the times, for example, “cloud era,” “big data,” letter words “yyds,” and “xdm.” Alternatively, words that do not exist in the dictionary may also be called new words or unregistered words [23].

New word can be judged as “new” from three aspects:

(1) Degree of solidification refers to the degree of closeness between words in a field. For example, words

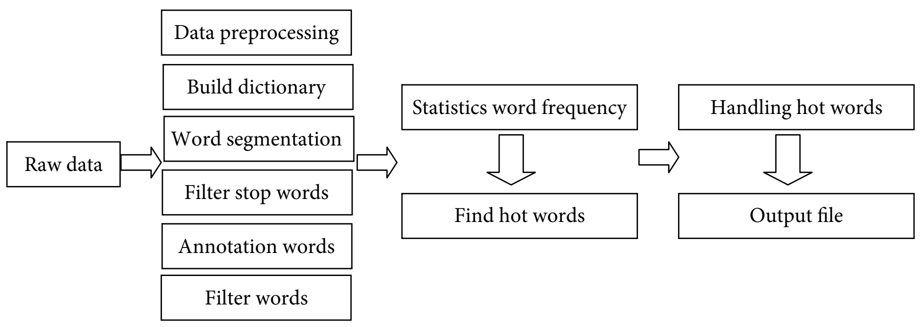


Figure 6: Process flow chart of mining hot words.

Table 3: Hot word list of online review text based on natural language processing.

Scenic area name	Hot words
A01	Animal, zoo, circus, project, scenic, queue, roller coaster, children, ticket, wonderful, train, park, feel, place, tour, and advice
A02	Fireworks, children, queue, animals, project, time, roller coaster, play, place, ocean, ocean kingdom, wonderful, scenic, park, facilities, and feel
A03	Project, tea valley, scenery, queue, place, scenic, Grand Canyon, attractions, feeling, time, roller coaster, small fire, tour, facilities, and environment
A04	Project, queue, ID, roller coaster, time, facilities, feel, place, a little, hours, night, haunted house, snowy, Halloween, and advice
A05	Attractions, feel, place, Shenzhen, architecture, world, guide, tickets, project, night, scenic, advice, tour, queue, scenery, and landscape
A06	Guangzhou, night view, Ferris wheel, queue, small barbarian waist, scenery, landscape, landmark, Pearl River, building, boarding tower, and elevator
A07	Attractions, place, folk village, scenic spot, China, feel, scenery, wonderful, program, ethnic, landscape, Shenzhen, tour, culture, night, and advice
A08	Animal, zoo, child, kids, tickets, feel, tiger, scenic, place, wonderful, time, small children, project, show, kind, and environment
A09	Scenic spot, attraction, scenery, view, place, climbing, time, feeling, tour, yang yuan mountain, mountain, and advice
A10	Game, project, animals, children, motorized, children, play, zoo, place, play, facilities, queue, a little, children, tickets, and feel
A11	Place, landscape, scenery, project, wedding photos, play, children, scenic, attractions, games, very beautiful, environment, wedding, and queue
A12	Fish, place, environment, scenery, attractions, play, garden, landscape, a little, park, Lingnan, feel, characteristics, architecture, scenic, and goldfish
A13	Hot spring, front desk, environment, service attitude, enthusiasm, waiter, soak hot spring, hot spring pool, fruit, attitude, pool, water, and staff
A14	Underground river, scenery, attractions, scenic spot, cave, queue, guide, cave, place, feel, scenery, tour, stalactites, very beautiful, and book tickets
A15	Animal, place, mermaid, feel, beluga, hour, penguin, little kids, fish, ocean, polar bear, show, and time
A16	Scenic spots, scenery, places, Guanyin, scenery, Nanhai, Foshan, environment, up the mountain, climbing, famous mountain, Xiqiao, Guangdong, and walking

such as “research” and “water cup” have a relatively high degree of solidification, while words such as “Haier” and “Gree” have a relatively low degree of solidification

(2) Degree of freedom refers to the degree to which a field can be used freely. For example, “Jujue” has the same degree of solidification as “Jejue,” but the freedom degree of “Jejue” is far less than that of “Jejue”

(3) The IDF (inverse document frequency) of new words is called the inverse document frequency of new words. If a word appears a high number of times in an article, it is most likely a new word. But if the word also appears a high number of times in the entire text corpus, then it may be a common word, not a new word

There are generally two approaches to new word discovery: rule-based new word discovery and statistics-based new

Table 4: The characteristics of scenic spots obtained by the new word discovery.

Scenic spots name	Scenic features
A01	Worlds of fun and online booking
A02	Motorized games, whale shark house, and penguin hotel
A03	Amusement facilities, mesa, tragen, motorized games, and online booking
A04	Direct swipe, Maya water, snowy eagles, rides, Snowy Mountain Flying Dragon, and water world
A05	Philadelphia tower, continental area, reduced version, itinerary, miniature view, and famous buildings
A06	Jumper, observation deck, landmark building, extreme skyscraper, landmark, Zhujiang New Town, world no. 1, and Haixinsha
A07	Miniature landscape and oriental neon
A08	Direct swipe, distance contact, and water park
A09	Changlao peak, Yang yuan stone, yin yuan stone, Danxia on water, Hosomei Zhai, and buy on site
A10	Motorized games, pick up tickets, suitable for a family, clip the doll, and clip the doll
A11	Wedding photography, with children, motor games, and wedding photography
A12	Pick up tickets, suitable for a family, and ancient fragrance
A13	Pick up tickets, no borders, and fruit drinks

Table 5: The characteristics of hotels obtained by the new word discovery.

Hotel name	The characteristics
H01	Near the metro, reception, next to East Station, old five star, free upgrade, full facilities, high train station, and clean and tidy
H02	Shennan avenue, convenient transportation, and full facilities
H03	Chang long, impressed, and great location
H04	Suitable for children and convenient transportation
H05	Free upgrade, easy access, platinum 5 stars, and clean and hygienic
H06	Suitable for strip, De Beer Plaza, and free upgrade
H07	Yutang spring, old five star, free upgrade, Sha Mian Park, hometown water, Servcorp house, toiletries, White Swan Lake, and redecoration
H08	Clean and tidy, fully equipped, free parking, front desk reception, close to West Lake, and great location
H09	Suitable for children, breakfast is plentiful, and clean and tidy
H10	Sea World, free upgrade, clean and tidy, check in, and decoration style
H11	Suitable for the strip, fairview, artificial beach, and breakfast buffet
H12	Apartment underground station, free upgrade, affordable, free parking, at the underground entrance, sound proofing, with character, and eating and shopping
H13	Changlong, lovers' road, breakfast is plentiful, and fully equipped

word discovery [24]. New word discovery based on rule can be performed by constructing a template for new word matching, and the obtained results have a relatively high accuracy rate. New word discovery based on statistics is to identify new words by counting the word frequencies in the corpus. This method is more portable and flexible and requires a certain model for training. In this paper, an algorithm based on mutual information and left-right entropy is used to discover new words. The specific steps are as follows:

(1) Stop word processing

The text that needs to be processed often contains many meaningless words. This paper replaces these texts or removes stop words to get cleaner texts.

(2) Use three thresholds to judge new words

(i) *Minimum Mutual Information*. The greater the mutual information, the higher the correlation. This paper uses the n-gram software for word segmentation and then calculates mutual information for these words. If it is below the threshold, it means that it cannot be a word

(ii) *Minimum Entropy*. The larger the entropy, the more abundant the neighboring words. This paper calculates the minimum value of left entropy and right entropy. If the minimum value is lower than the threshold value, it means that the word cannot be formed

(iii) *Minimum Number of Occurrences*. If the number of occurrences of a word is less than the set minimum number of occurrences, it is filtered out

(3) Mining of characteristic in scenic spots and hotels

In this paper, the algorithm based on mutual information and left-right entropy is used to discover new words. At the same time, this algorithm also integrates the Python module of SmoothNLP. The implementation of SmoothNLP is different from the work of mutual information and left-right entropy. For the same text, the new words discovered by the two methods are not consistent. Therefore, we calculate the two methods together and obtain the common vocabulary of the two methods. In addition, natural language processing also provides an option "whether to extract only words that are not in the dictionary." Thereby, the search scope can be expanded to find more unregistered new words. Therefore, this pattern is also incorporated into the method of new word discovery. Here, these three methods are combined to discover new words in the online review texts by adjusting the parameters. This new word discovery method considers both the breadth and novelty of new words and the acceptability of new words, which can reflect the characteristics of scenic spots or hotels. Finally, the test is carried out on the online review text with high comprehensive evaluation, and the experimental results show: Table 4 shows the characteristics of 13 scenic spots obtained by the new word discovery algorithm. Table 5 shows the characteristics of the 13 hotels obtained by the new word discovery method.

7. Conclusion

At present, it is a research hotspot in the field of tourism management to provide directions for the development of tourism by the mining of online review texts. This paper first analyzes the research status and finds that the current application of online review texts is not systematic, and there is a lack of a comprehensive scoring mechanism for the service quality of scenic spots and hotels. Then, this paper proposes a novel mining and application of tourism online review text based on natural language processing and text classification technology. A series of new methods are proposed here.

The first is to remove the invalid online review text and keep the valid text. The research focus of this paper is to propose a comprehensive evaluation of scenic spots and hotels based on text classification technology and sentiment analysis methods. This evaluation system can establish evaluation indicators for comprehensive scoring and select top-ranked scenic spots and hotels. Then, for the online review texts of high-quality scenic spots, this paper proposes to use natural language processing to mine hot words. The obtained hot words can intuitively reflect the impression of the scenic spot to tourists. Then, new words are discovered based on mutual information and left-right entropy methods. The service characteristics of high-quality scenic spots and hotels are excavated from new words. Finally, the above series of methods are tested on the massive tourism online review texts. The experimental results show that the new comprehensive evaluation method proposed in this paper can effectively select high-quality scenic spots and hotels. Hot words and new words can be efficiently mined from relevant online review texts. These methods can feedback tourism impressions from various aspects and levels and provide important basis for the development of tourism. Due to the limitations of the selected online review texts, the comprehensive evaluation method in this paper has certain regional characteristics and is not suitable for all tourist locations. There are two aspects of future research work. The first is to expand the crawling scope of online review texts as much as possible and establish more comprehensive evaluation indicators. The second is to consider how to use the discovered hot words and new words to provide personalized intelligent services for tourists and tourism practitioners.

Data Availability

No data were used to support this study.

Conflicts of Interest

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

Acknowledgments

This work was supported by the National Natural Science Funds of China (61272015) and the 2022 Henan Province Science and Technology Research Project: "Construction and Application of Intelligent Ontology in Internet of Things Based on Semantic Concept Model" (222102210316).

References

[1] W. Chen, Z. Xu, X. Zheng, Q. Yu, and Y. Luo, "Research on sentiment classification of online travel review text," *Applied Sciences*, vol. 10, no. 15, p. 5275, 2020.

[2] X. Li, R. Law, G. Xie, and S. Wang, "Review of tourism forecasting research with Internet data," *Tourism Management*, vol. 83, no. 4, article 104245, 2021.

[3] W. Yuan, X. Xin, and F. Xuegang, "Research on tourist' percieved image of ancient town using web text mining methods: a case study of Zhujiajiao," *Tourism Science*, vol. 27, no. 5, pp. 86–94, 2013.

[4] C. A. I. Yi, Y. A. N. G. Yang, and Y. Hongmei, "Research on audience's perception of tourism brand of Guiyang based on the text mining of ROST," *Journal of Chongqing Normal University(Natural Science)*, vol. 32, no. 1, pp. 126–134, 2015.

[5] W. Yaobin, Y. Ling, and S. Chuanling, "Comparative research on travel sharing of typical travel website based on text mining-taking Gansu Province as an example,"

Resource Development & Market, vol. 33, no. 1, pp. 100–104, 2017.

[6] L. Ping, C. Tian, and W. Fuyuan, "Urban tourism community image perception and differentiation based on online comments: a case study of Beijing," *Geographical Research*, vol. 36, no. 6, pp. 1106–1122, 2017.

[7] L. I. N. Xuanmiao, "A comparative study of tourists' opinions and suggestions on Qinbeifang of Guangxi based on text mining," *Market Forum*, vol. 7, pp. 69–74, 2017.

[8] H. Ming, S. Jianjun, and C. Ying, "Text classification based on Naive Bayes: a review," *Information Science*, vol. 34, no. 7, pp. 147–154, 2016.

[9] L. Lv, Z. Wu, L. Zhang, B. B. Gupta, and Z. Tian, "An edge-AI based forecasting approach for improving smart microgrid efficiency," *IEEE Transactions on Industrial Informatics*, vol. 3, pp. 1–1, 2022.

[10] L. Lv, Z. Wu, J. Zhang, Z. Tan, L. Zhang, and Z. Tian, "A VMD and LSTM based hybrid model of load forecasting for power grid security," *IEEE Transactions on Industrial Informatics*, vol. 11, p. 1-1, 2021.

[11] G. U. A. N. Lei and S. U. N. Tao, "An efficient parallel and distributed solution to nonconvex penalized linear SVMs," *Frontiers of Information Technology & Electronic Engineering*, vol. 21, no. 4, pp. 587–603, 2020.

[12] L. Zhang, Z. Huang, W. Liu, Z. Guo, and Z. Zhang, "Weather radar echo prediction method based on convolution neural network and long short-term memory networks for sustainable e-agriculture," *Journal of Cleaner Production*, vol. 298, article 126776, 2021.

[13] Y. Lei Zhang, Q. G. Huo, Y. Ma, Q. Liu, and W. Ouyang, "A privacy protection scheme for IoT big data based on time and frequency limitation," *Wireless Communications and Mobile Computing*, vol. 2021, 10 pages, 2021.

[14] Y. Ye, Q. Wu, Y. Li, K. P. Chow, L. C. Hui, and S. M. Yiu, "Unknown Chinese word extraction based on variety of overlapping strings," *Information Processing & Management*, vol. 49, no. 2, pp. 497–512, 2013.

[15] L. Zhang, C. Xu, Y. Gao, Y. Han, X. du, and Z. Tian, "Improved Dota2 lineup recommendation model based on a bidirectional LSTM," *Tsinghua Science and Technology*, vol. 25, no. 6, pp. 712–720, 2020.

[16] L. Zhang, S. Tang, and L. Lv, "An finite iterative algorithm for sloving periodic Sylvester bimatrix equations," *Journal of the Franklin Institute*, vol. 357, no. 15, pp. 10757–10772, 2020.

[17] L. Lv, J. Chen, Z. Zhang, B. Wang, and L. Zhang, "A numerical solution of a class of periodic coupled matrix equations," *Journal of the Franklin Institute*, vol. 358, no. 3, pp. 2039–2059, 2021.

[18] C. Nan, Z. Jian, and W. Juqing, "Not only rating, but also text content will produce an influence on review helpfulness," *Luojia Management Review*, vol. 1, pp. 15–26, 2014.

[19] D. Jianyong, G. Huijuan, and Z. Mei, "Research on the method of network review extraction," *Journal Of North China University*, vol. 27, no. 1, pp. 7–12, 2015.

[20] L. Lv, S. Tang, and L. Zhang, "Parametric solutions to generalized periodic Sylvester bimatrix equations," *Journal of the Franklin Institute*, vol. 357, no. 6, pp. 3601–3621, 2020.

[21] D. Brzezinski and J. Stefanowski, "Prequential AUC: properties of the area under the ROC curve for data streams with concept drift," *Knowledge and Information Systems*, vol. 52, no. 2, pp. 531–562, 2017.

[22] Q. Li, S. Li, S. Zhang, J. Hu, and J. Hu, "A review of text corpus-based tourism big data mining," *Applied Sciences*, vol. 9, no. 16, p. 3300, 2019.

[23] A. Usai, M. Pironti, M. Mital, and C. Aouina Mejri, "Knowledge discovery out of text data: a systematic review via text mining," *Journal of Knowledge Management*, vol. 22, no. 7, pp. 1471–1488, 2018.

[24] L. Weitong, L. Peiyu, and L. Wenfeng, "New word discovery algorithm based on mutual information and branch entropy," *Application Research of Computers*, vol. 36, no. 5, pp. 1294–1296, 2019.

Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 7245403, 11 pages
https://doi.org/10.1155/2022/7245403

Research Article

Automatic Detection of Android Malware via Hybrid Graph Neural Network

Chunyan Zhang,[1] **Qinglei Zhou**,[2] **Yizhao Huang**,[1] **Ke Tang**,[1] **Hairen Gui**,[1] **and Fudong Liu**[1]

[1]*State Key of Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, 450001 Henan, China*
[2]*College of Computer Information and Engineering, Zhengzhou University, Zhengzhou, 450001 Henan, China*

Correspondence should be addressed to Fudong Liu; lwfydy@126.com

Received 7 March 2022; Accepted 19 April 2022; Published 11 May 2022

Academic Editor: Lei Zhang

Automatic malware detection was aimed at determining whether the application is malicious or not with automated systems. Android malware attacks have gained tremendous pace owing to the widespread use of mobile devices. Although significant progress has been made in antimalware techniques, these methods mainly rely on the program features, ignoring the importance of source code analysis. Furthermore, the dynamic analysis is low code coverage and poor efficiency. Hence, we propose an automatic Android malware detection approach, named HyGNN-Mal. It analyzes the Android applications at source code level by exploiting the sequence and structure information. Meanwhile, we combine the typical static features, permissions, and APIs. In HyGNN-Mal, we utilize a deep traversal tree neural network (Deep-TNN) to process the code structure information. Particularly, we add position information to code sequence information before putting in self-attention mechanism. The evaluations conducted on multiple public datasets indicate that our method can accurately identify and classify the malicious software, and their best accuracy is 99.62% and 99.2%, respectively.

1. Introduction

Android, the most popular mobile operating system, has attracted millions of users around the world. The market share of Android smartphones was 70.69 percent in April 2020 [1], and there were no signs of a decrease in the near future [2]. The global dominance of Android operating system and the rich data storage by smartphones make Android users an attractive target for cyberattackers. Kaspersky's mobile malware (a short form used for malicious software) evolution showed that about 3.5 million malicious installation packages appeared in 2019 [3] and the number of new Android malware in 2020 grew by 2.1 million, breaking the established trend of continuous decline [4]. Usually, malware developers use obfuscation methods to generate malware variants which lead to misjudgments and omission of detection results. Therefore, it is an urgent and important task for researchers to study the accurate automatic detection of Android-based malware.

The malware detection techniques mainly fall into three approaches: (1) static analysis obtains Android features by scanning the decompiled Android package's files, permissions, source code, and API calls. (2) Dynamic analysis observes and selects features form the system calls, read and write operations, and network data when running the application in a closely monitored virtual environment. (3) Hybrid analysis is the combination of dynamic and static analysis. Many existing Android malware detection approaches have reported excellent results [5–13]. For example, Kim et al. [5] built malware detection models based on seven kinds of static features such as string feature, opcode feature, and API feature, while Fan et al. used subgraphs as sample features to process Android malware familial classification [6]. There are lots of significant research results [14–21] detecting Android malware adopt hybrid analysis. For example, Wong and Lie [21] proposed a hybrid system, named IntelliDroid, which was generic Android input generator for the analysis of any Android application. The dynamic analysis is effective against all types of malwares. Tam et al. [22] built a dynamic analysis system, CopperDroid, to extract interactions between the app and the operating system, as well as interprocess and in-process communications. CopperDroid successfully disclosed more than 60% malicious behaviors of 1365 malware samples. The method has a limitation that it can only automatically check parameters of methods contained in the interface that owns the AIDL file [23]. Alzaylaee et al. [24] adopted a DL solution via dynamic analysis to detect malicious Android apps, the method used a real Android device and 31,125 apps for the experiments and achieved an accuracy of 0.952.

Currently, most of the previous studies ignoring in-depth analysis of application, such as the structure information, interdependency of program, and time complexity. Particularly, the dynamic analysis is accurate but time-consuming and labor-intensive [25]. The hybrid analysis methods can generate a better overall picture of the program but more complex and time-consuming [26]. The application of deep learning (DL) techniques on malware detection is effective [27–32]. Talha et al. [30] suggested a permission based on malware identification system, which analyzed the permissions requested by the application for identifying whether it is malicious or not. Similar work, Cen et al. [31] analyzed the code level of Android malware apps and trained a probabilistic classifier system to predict whether an application is malicious or not. Alazab M et al. [32] proposed permissions and frequency analysis of API calls, and their approach can determine the similarities among malware families. They achieved an F-measure of 0.943 on the dataset of 27,891 apps. Besides, a typical malware detection approach is to convert the disassembled malware codes into a graph representation, such as a grayscale image [33, 34], control-flow graph (CFG), and data flow graph (DFG) and then use machine learning algorithms to identify the malware. Specifically, the DL methods are often more suitable to capture the semantic knowledge within Android apps than the traditional machine learning methods [35]. Considering the large number of datasets in reality, our method follows the static analysis, which does not depend on the compiler and the environment of executed program. Furthermore, we incorporate joint-feature vectors into the hybrid model and evaluate the model on the public datasets. In this way, we can fully utilize the syntactic and semantic information of an Android app without dataset bias that affects the experimental result.

Therefore, we present a malware detection approach based on hybrid graph neural network, which is designed for identifying and detecting the Android malware. We extract semantic information of Android application from source code and program levels. At the source code level, we propose an effective analysis way (see Section 3 for details) inspired by natural language processing (NLP) technology. At the program level, we use the Androguard tool to extract the typical features, permissions, and APIs. The Android applications are represented by combining these three semantic vectors to address the Android malware detection issue.

The main contributions of this paper are as follows:

(i) We propose a new automatic Android malware detection approach, HyGNN-Mal, which can accurately identify the malware and its type via a hybrid graph neural network

(ii) We subjoin the row and column information to source code sequence, which enhances the semantic of Android application

(iii) We use abstract syntax tree (AST) to represent the structure information of source code. To the best of our knowledge, this the first work that uses AST to analyze the Android applications

(iv) We verify the effectiveness of HyGNN-Mal on multiple real-word public datasets to avoid data bias, and the experimental results show a superior accuracy compared to existing methods

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 presents our approach in detail. Section 4 reports and analyzes experimental results. Finally, Section 5 gives the paper conclusion and future trend discussion.

2. Related Work

2.1. Deep_TNN (Deep Traversal Tree Neural Network). Deep traversal neural network (Deep_TNN) is a graph model structure that we proposed to traverse the AST in this paper. Different from images (a set of pixels) and natural language (a sequence of words), graph data is more complex that there are at least two types of information in a graph: nodes and the edges. Hence, we build a suitable graph neural network architecture, Deep_TNN, which can be viewed as a variant of graph convolutional network (GCN) [36].

Suppose a graph $G = (V, E)$, where the vertex set $V = \{(v_1 \cdots v_i \cdots v_N) | 1 \le i \le N\}$ and N is the number of vertices. Let the number of features for each vertex be D, and then, the vertex feature of G is a vector $X = N \times D$. $E = \{(v_i, v_j) | i, j \in N\}$, where E represents the relationship among vertices and then forms an adjacency matrix $A = N \times N$. A and X are the input of GCN, and the message transmission between GCN layers is as follows:

$$H^{(l+1)} = \delta\left(\tilde{D}^{-1/2} \cdot \tilde{A} \cdot \tilde{D}^{-1/2} \cdot H^{(l)} \cdot W^{(l)}\right), \tag{1}$$

where $\tilde{A} = A + I$ and I is an identity matrix and $\tilde{D}$ represent the degree matrix of GCN. The diagonal elements are the degrees of each vertex of $\tilde{A}$; for example, the degrees of v_i is the number of edges in G connected to v_i, H is the feature vector of each GCN layer, for the input layer, H is X, and δ is an nonlinear activation function.

2.2. Self-Attention Mechanism. Self-attention [37] is a feature extractor layer, which allows the inputs to interact with each other and find out the areas where they should pay more attention. Superior than RNN, self-attention mechanism can capture long-distance interdependence and calculate in parallel. Here is an introduction to the self-attention mechanism.

Let $X = \{(x_1 \cdots x_i \cdots x_n) | 1 \le i \le n\}$ as one input vector of self-attention, and the corresponding output is $H = \{(h_1 \cdots h_i \cdots h_n) | 1 \le i \le n\}$, where n is the number of features for X. x_i has three different vectors, query (Q), key (K), and value (V), and the calculation formulas are as follows:

$$\begin{aligned} Q &= X \cdot W^Q, \\ K &= X \cdot W^K, \\ V &= X \cdot W^V, \end{aligned} \tag{2}$$

where the W^Q, W^K and W^V are the trainable parameter matrices. The output of each x_i can be calculated as follows:

$$h_i = \sum_{j=1}^{n} \text{soft max}\left(\text{simlarity}\left(q_i, k_j\right)\right) \cdot v_j, \tag{3}$$

where $q_i \subset Q$, $k_j \subset K$, $v_j \subset V$, and $i, j \in (1, n)$. Yet, the weights of self-attention mechanism only depend on the relativity between q_i and k_j, ignoring the position information of X.

2.3. Bidirectional Gated Recurrent Unit. GRU is a variant of long short-term memory (LSTM). The main difference between LSTM and GRU lies in that GRU uses an update gate and a reset gate and LSTM uses forget gate, input gate, and output gate [38–40]. The final GRU model is more concise and effective in long-sequence applications. A bidirectional GRU (Bi-GRU) is a combination of two GRUs in opposite directions.

Let $X = \{(x_1 \cdots x_i \cdots x_n) | 1 \le i \le n\}$ as an input vector of Bi-GRU, and the corresponding output is $H = \{(h_1 \cdots h_i \cdots h_n) | 1 \le i \le n\}$, where n is the number of features for X. Take one input at time t for example, x^t.

$$z^t = \delta\left(W_z \cdot x^t + U_z \cdot h^{t-1} + b_z\right), \tag{4}$$

$$r^t = \delta\left(W_r \cdot x^t + U_r \cdot h^{t-1} + b_r\right), \tag{5}$$

$$\tilde{h}^t = \tanh\left(W_h \cdot x^t + U_h \cdot \left[x^t, h^{t-1}\right] + b_h\right), \tag{6}$$

$$h^t = \left(1 - z^t\right) \cdot h^{t-1} + z^t \cdot \tilde{h}^t, \tag{7}$$

where z^t and r^t represent the update gate and reset gate. W_z, W_r, W_h, U_z, U_r, and U_h are all weight matrices. b_z, b_r, and b_h are all bias vectors. $\tilde{h}^t$ is the current hidden layer information. h^t is the final result.

3. Approach

Figure 1 shows typical data-driven workflow. It contains raw Android APK file collection, the feature extraction and representation of application, malware detection modelling, and model evaluation. According to this workflow, we illustrate the implementation of HyGNN-Mal in Figure 2.

3.1. Problem Definition. Given one Android app A_i, we set a constant label Y for it to indicate whether is benign or malicious, even the type of malware, where $Y_i \in \{\text{benign}, \text{trojan}, \text{adware}\}$. Then, for a set of known labels, we can build a training set $T = \{(A_i, Y_i)\}$. We aim to train a deep learning model for learning a function f that maps an Android app A to a feature vector V, so that for any Android app, based on the similarity score $s_{ij} = f(A_i, Y_j)$, we can determine which label is most likely to belong to A_i.

3.2. Hybrid Model. The process of HyGNN-Mal is as follows. In the training stage, a set of Android apps with labels are processed by three separate parallel branches, self-attention, Deep_TNN, and Bi-GRU. We merge the Seq_vector, AST_vector, and Behavior_vector into one APP_vector to represent the Android app and identify whether the app is malicious or not by MLP. In the testing stage, we input a set of Android apps without labels to HyGNN-Mal; if the app is malicious, return its type, else return benign. The training and testing workflow is shown in Figure 2.

3.3. Feature Extraction. We extract the features of Android apps from both source code and program levels. For source code level, we get the source code of Android apps by reversing engineer APK files. For program level, we use the Androguard tool to extract the typical features, permissions, and APIs.

3.3.1. Source Code Analysis. As we all know, one application (also named program) contains multiple .java files, while each .java file contains multiple functions. We analyze the function-level source code in two ways, source code sequence and its parsed AST, as shown in Figure 3.

On the one hand, we use the javalang package of python to parse the function code f into abstract syntax tree (AST). According to the production rules in javalang, the AST nodes are divided into three types: string (terminal nodes), set ("Modifier") and class name. Terminal nodes refer to the identifier tokens in f, and the nonterminal nodes represent syntactic structure of f. Particularly, we incorporate some edge types to exploit useful information of AST. (1) Next_TNode: connect a terminal node to the next terminal node, which just shows the order of f. For example, the purple connection "Public-static-ForDisplay" in the red box (aa) of Figure 3, corresponds to the edges (Public and static) and (static and ForDisplay). (2) Next_BNode: connect a node to its next brother node (from left to right), which is solving the problem that graph neural networks do not consider the order of nodes. For example, the green connection "Public-static" and "Modifier-ForDisplay," correspond to the edges (Public and static) and (Modifier and ForDisplay). (3)

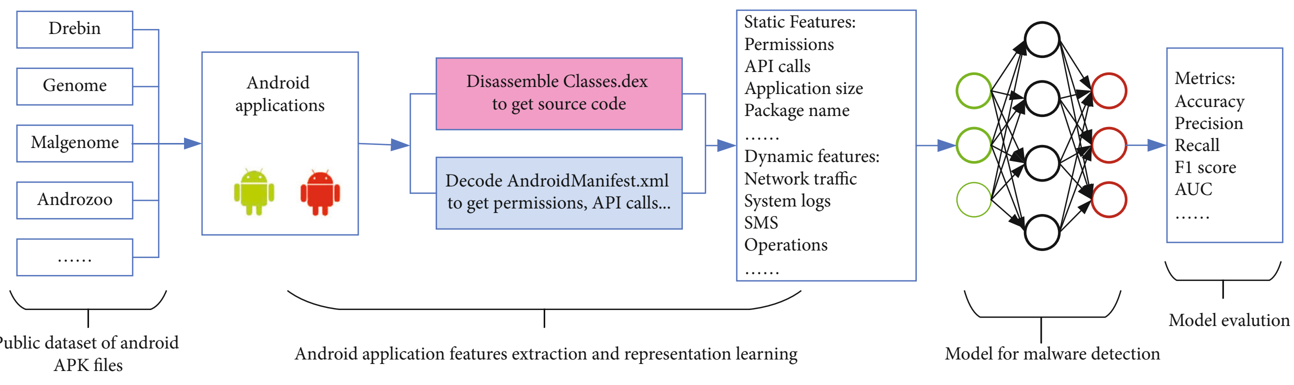


Figure 1: A typical data-driven workflow of Android malware detection.

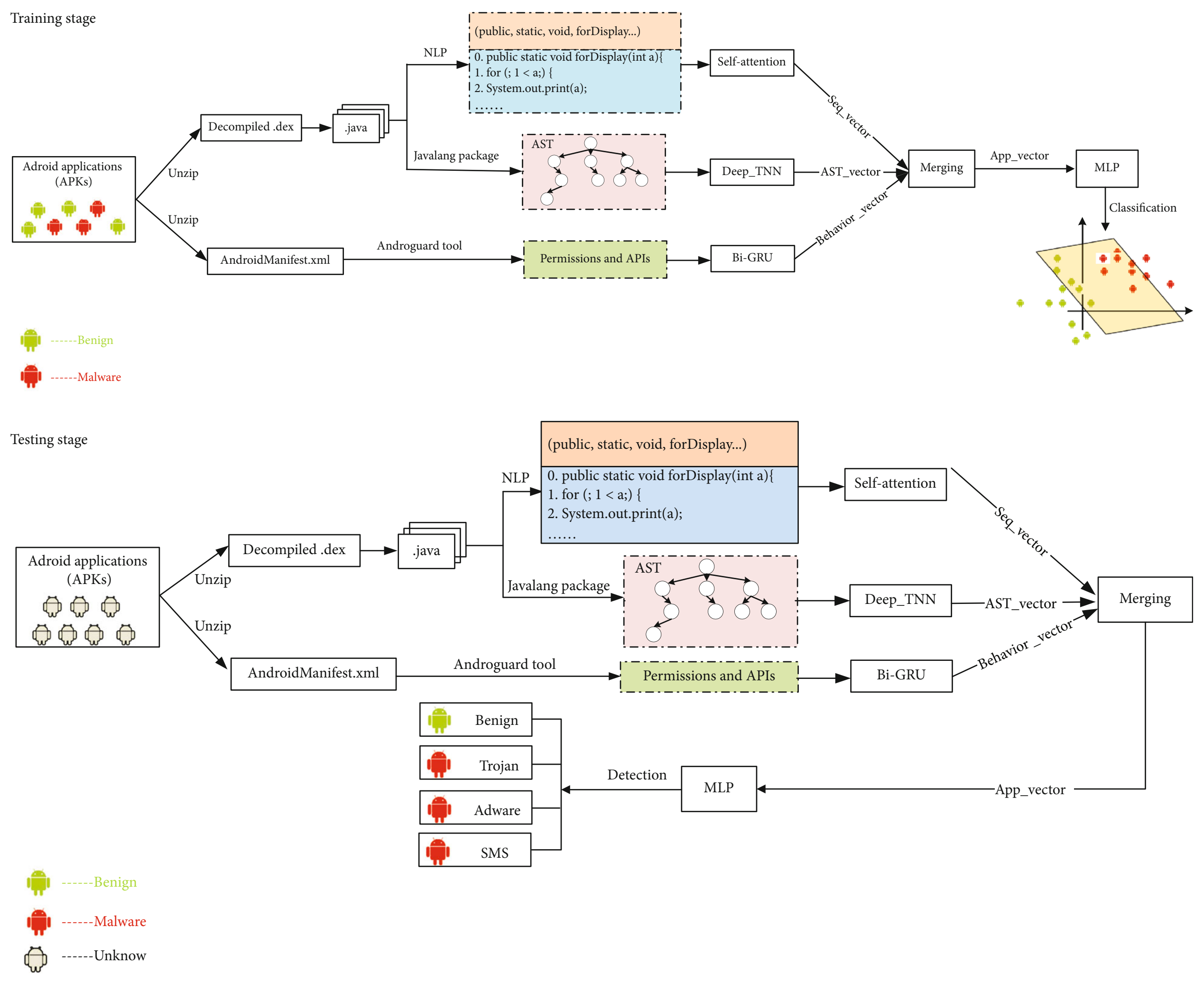
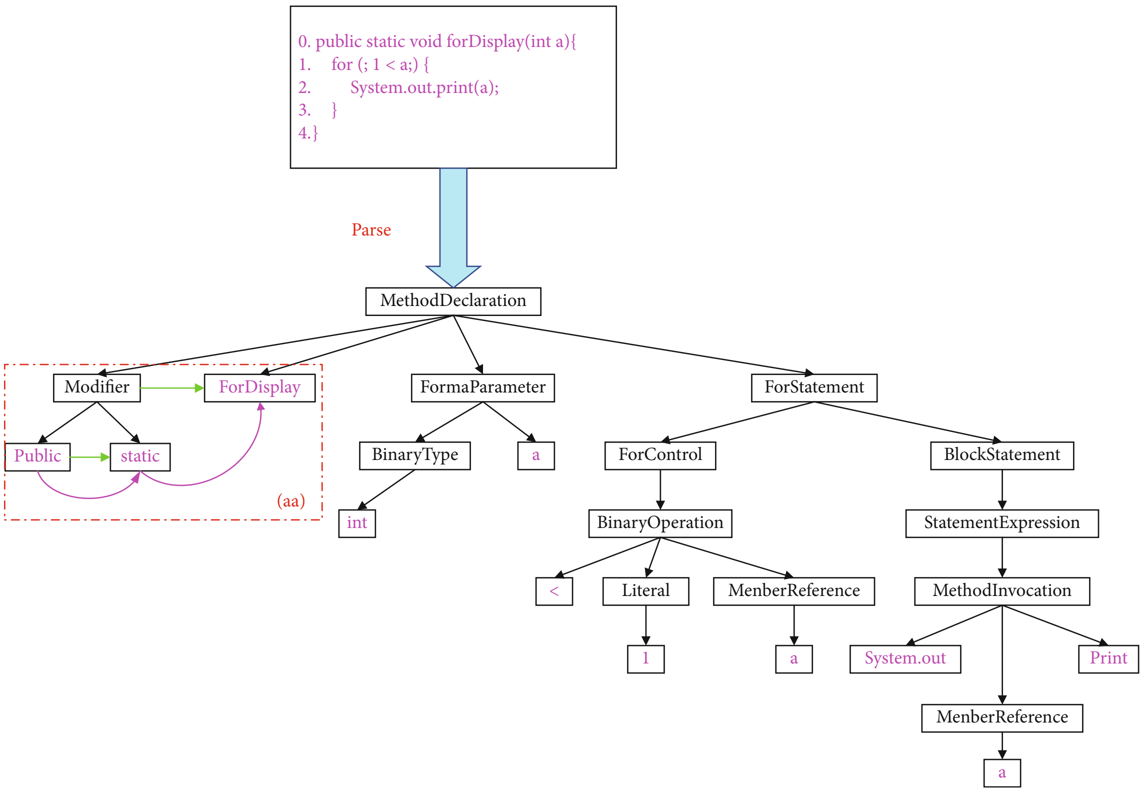


Figure 2: The overview of HyGNN-Mal (training stage and testing stage).

Building a directed graph based on AST. Because the effect of directed graphs is similar to undirected graph, but the number of it is half less than undirected graphs, so time efficiency can be greatly improved. Besides, the directed graph can represent structural information more accurately. At last, we apply depth-first traversal to get the vertex set v and edge set e of AST, where $v = (t_1, t_2 \cdots t_m)$, m is the number of function code tokens; $e = (e_1, e_2 \cdots e_n)$, n is the number of edges; $e_{ij} = (t_i, t_j)$; and t_i, t_j indicate the start node and end node of each edge e_{ij}. After embedding the v, we input it into Deep_TNN shown in Figure 4 for feature extraction and finally get the vector of v, named $AVec$.



```
public static void forDisplay(int a){
    for (; 1 < a;) {
        System.out.print(a);
    }
}
```

Figure 3: An example of function-level source code analysis: sequences and AST.

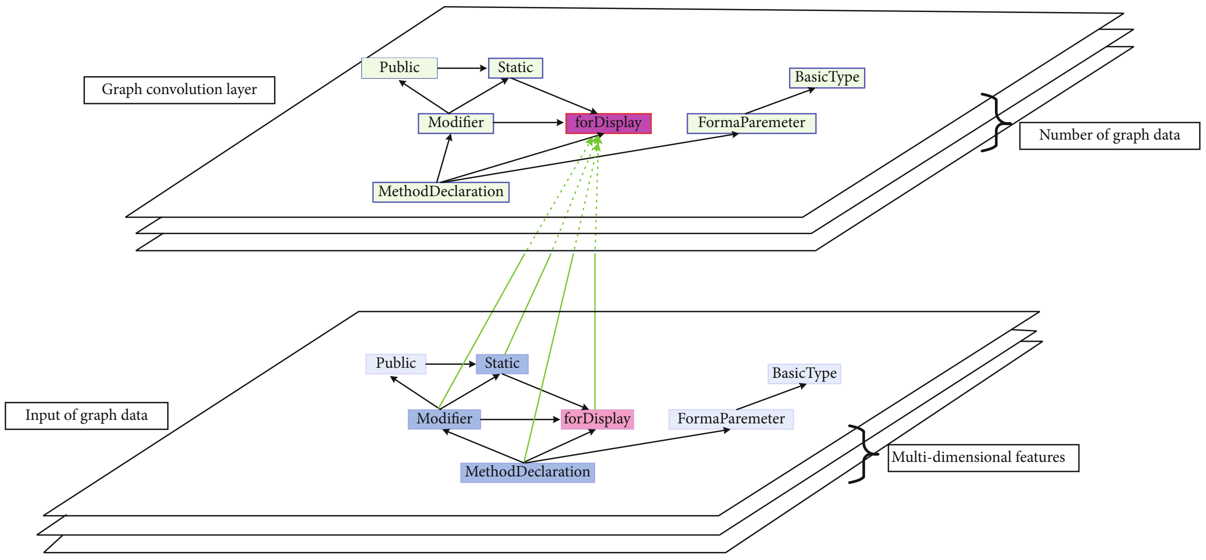


Figure 4: Deep_TNN to extract features of AST.

On the other hand, we add row and column position information to the source code, as shown in Figure 5. It just corrects the self-attention mechanism shortcomings mentioned in Section 2.3. First, we change the source code to a plain text sequence and assign a column position value to each token, as shown in the blue box. Second, we assign a row position value to each line of the source code, as shown in the red box. At last, we combine the embedding of token lexical information and two additional position information (row and column) mentioned above into a feature vector and input the self-attention mechanism to train for a new feature vector, named $SVec$; the workflow is shown in Figure 6. For example, in the word "static," the row position information is "0," and the embedding is Emb_1. Meanwhile, the column position information is "1," and the embedding is Emb_2. Moreover, the lexical information is

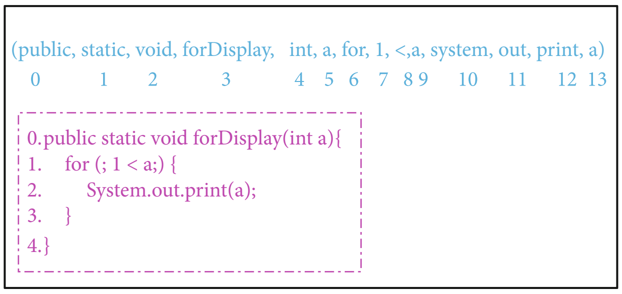

(public, static, void, forDisplay, int, a, for, 1, <,a, system, out, print, a)
0 1 2 3 4 5 6 7 8 9 10 11 12 13
0.public static void forDisplay(int a){
1. for (; 1 < a;) {
2. System.out.print(a);
3. }
4.}

Figure 5: Row and column position information of source code.

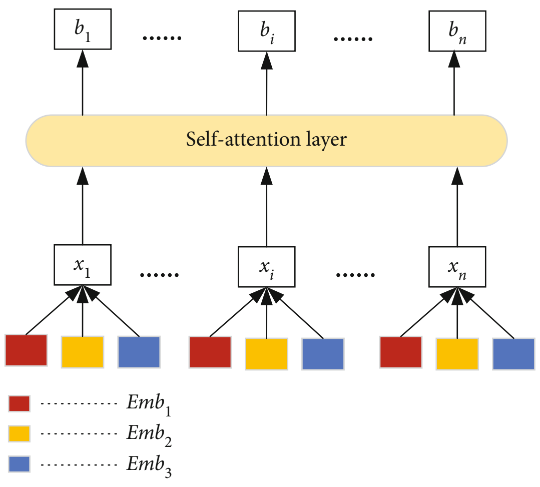


Figure 6: Self-attention mechanism to extract features of sequences.

Emb_3; thus, the final embedding of "static" is $Emb = cat(Emb_1, Emb_2, Emb_3)$; after the self-attention, we can get the vector of "static."

3.3.2. Program Analysis. The Android APK was analyzed by Androguard tool, including the APK object, DEX file object, and analysis object. In most cases, a malware typically applies for more permissions than a normal software and tends to request a special set of permissions. APIs can effectively reflect the malicious behaviors to a certain extent. We get the permission sequences from APK object and the API sequences from analysis object. Each APK is represented as {label, permissions list, API sequences list}, which is input into the Bi-GRU to get the behavior feature, $BVec$. The process is shown in Figure 7.

Finally, by merging $AVec$, $SVec$, and $BVec$, we get the feature vector of f, $fVec = cat(AVec, SVec, BVec)$, where $fVec$ is a matrix $M^{N \times D}$; N is the number of features, and D is the dimension of each feature. In this way, we represent an application as a feature vector full of semantic information, just as $Vec = (fvec_1, fvec_2 \cdots fvec_k)$, where k is the number of functions in the application. The HyGNN-Mal uses Vec to realize the automatic detection of Android malware through the full connection layer.

4. Experiments

4.1. Hardware Details. The hardware on which we implemented, trained, and tested our model included two Intel(R) Xeon(R) Gold 6154 CPU @ 3.00 GHz, 128 GB RAM, and two Titan XP GPUs. It was necessary to train on GPUs with 64 GB VRAM due to the large size of our model.

4.2. Datasets. In order to avoid the experimental effect bias caused by dataset construction, we select multiple samples from the public data source (see Table 1). In malware binary classification study, we download 16,000 malicious applications from the Drebin (DB) [41] and AndroZoo (AZ) (AppChina and Genome) [42] and the same number of benign applications from Google Play (GP) of AZ. In multiclassification study, we consider a new Android malware dataset, CICMalDroid (CICMD) 2020 [25], which is intentionally spanning between five distinct categories (adware, banking malware, SMS malware, riskware, and benign), and then download 15,000 samples from it. Each sample corresponds to an APK file, after the feature engineering extraction, the final remaining Android APKs. Furthermore, to avoid the potential biases introduced by our random sampling, we have removed the same samples to ensure that there are no overlap samples in our classification study.

In the following experimental tasks, we randomly split the total data of DB and AZ datasets into a training set and a test set with the ratio of 8 : 2. The same segmentation method is also used to CICMD dataset.

4.3. Performance Measures. The metrics are accuracy, recall, precision, and F1 score which have been evaluated to verify the effectiveness of HyGNN-Mal model.

(1) True positive (TP): the number of samples classified as malicious correctly

(2) False positive (FP): the number of samples classified as malicious wrongly

(3) True negative (TN): the number of samples classified as benign correctly

(4) False negative (FN): the number of samples classified as benign wrongly

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{8}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{9}$$

$$F1\ \text{score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{10}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{11}$$

4.4. Results. We optimize the performance of HyGNN-Mal by tuning variable hyperparameters. The final parameter settings used for all experiments are shown in Table 2. The Deep-TNN layer means the number of graph convolution

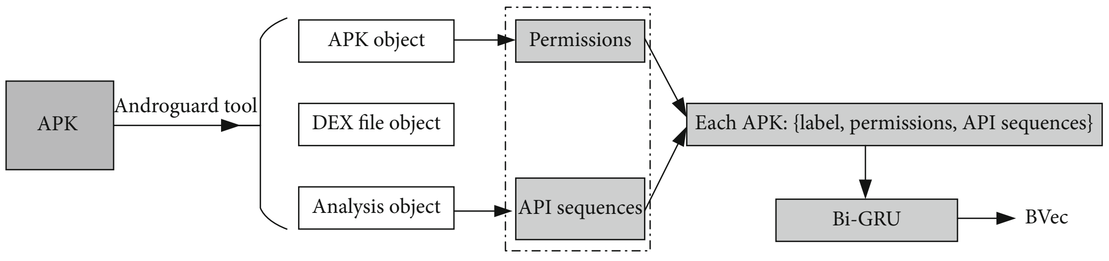


Figure 7: Bi-GRU to extract features of permissions and APIs.

Table 1: Dataset overview.

Dataset		Period	Download	Task	Available	Total available
DB		2010-2012	Malicious: 1,000	Binary classification (AZ and DB)	Benign: 11,765 Malicious: 11,820	23,585
AZ	AppChina Genome GP	2010-2019	Malicious: 15,000 Benign: 16000			
CICMD		2017-2018	15000	Multiclassification (CICMD)	1,685 (benign) 1,221 (adware) 2,008 (banking) 3,822 (SMS) 2,416 (riskware)	11,152

Table 2: Hyperparameter setting.

Parameters	Values
Embedding dimension	200
Deep-TNN layer	2
Deep-TNN filter	50
Dropout rate	0.1
Learning rate	0.001
Hidden layer dimension	625
Optimizer	Adam
Loss function	Cross-entropy loss
Batch size	128
Epoch number	40

layers. The Deep-TNN filter indicates the number of output channels in a graph convolution layer. The dropout rate is used to overcome overfitting by randomly excluding nodes during training. The batch size represents the number of samples selected for one training. The epoch means the number of iterations for the model training.

For the purpose of verifying the effectiveness of HyGNN-Mal, we conduct a number of experiments. These experiments answer the following research questions.

RQ1. What is the performance of HyGNN-Mal compared to the other well-performing methods?

RQ2. Why the HyGNN-Mal model we proposed performs well?

RQ3. What is the impact of source code level analysis on malicious code detection?

4.4.1. Comparison with Different Methods. We compare our methods with four well-performing methods, and the experimental results are shown in Table 3. The reason we choose these benchmarks is that the datasets or models they used are very similar to ours. It is suitable to highlight the effectiveness of the HyGNN-Mal by comparison.

Marcheggiani and Titov [43] showed that bidirectional LSTM (Bi-LSTM) and syntax-based GCNs have complementary modeling power, when GCNs were stacked on top of LSTM layers. Haq et al. [44] leveraged convolutional neural network (CNN) and Bi-LSTM to efficiently identify the persistent malware. They conducted comparative experiments on hybrid DL-driven architectures and DL benchmarks using publicly available datasets. Pei et al. [11] proposed AMalNet for Android malware detection and family attribution. They applied the GCNs and independently recurrent neural network (IndRNN) to fully consider the semantic distribution information of malware, such as character, word, and lexical feature. The above methods performed well, but they still fall short in application semantic extraction. We implemented the models (Bi-LSTM+GCNs, CNN+Bi-LSTM, and GCNs+IndRNN) to extract features described in Section 3.3. Mirzaei et al. [45] used a fast graph-mining algorithm to extract the ensembles of API calls and modeled them as Markov chains to construct the feature vectors for apps. They also utilized machine learning algorithms for classification. Furthermore, the paper [45] has opened their source code so that we can easily reproduce them on our dataset.

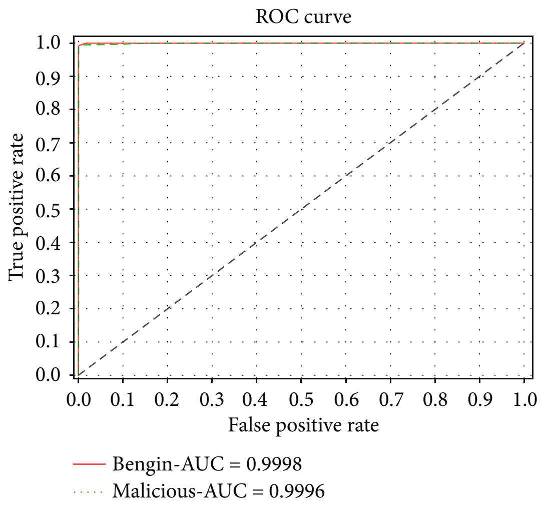
From Table 3, the results of [11, 43] are close to our method. However, they ignored the structural information of source code, while we use the AST and the location information of source code to represent in the paper. We also consider the behavioral characteristics of malware, permissions and APIs. The comparison indicated that HyGNN-Mal proposed in this paper surpassed previous efforts.

Table 3: Experimental results compared with benchmarks.

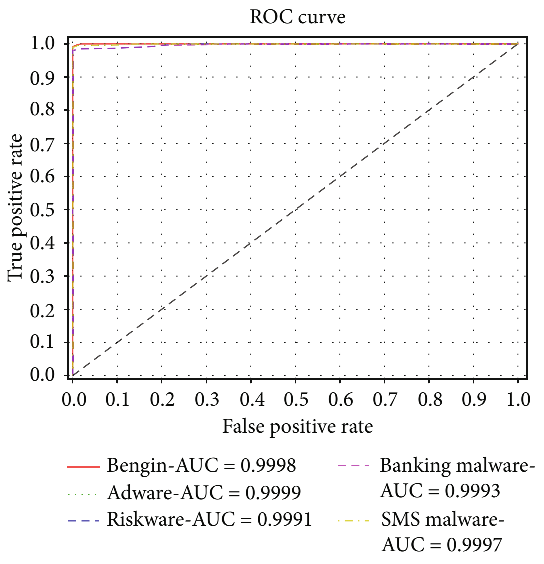
Method	Year	Binary-classify				Five-classify			
		Acc	P	R	F1	Acc	P	R	F1
Bi-LSTM+GCNs [43]	2017	99.23	99.54	99.11	99.32	98.36	98.10	98.42	98.25
AndrEnsemble [45]	2019	96.40	96.35	97.12	96.73	95.30	95.12	96.01	95.56
IndRNN+GCNs [11]	2020	99.15	98.82	99.21	99.01	98.50	98.42	98.30	98.36
Bi-LSTM+CNN [44]	2021	98.68	99.43	99.25	99.33	98.02	97.65	98.44	98.04
Bi-GRU+Deep_TNN+Self-attention	—	99.62	99.67	99.50	99.58	99.20	99.15	98.89	99.01

Table 4: Experimental results of different models.

Models	Binary-classify				Five-classify			
	Acc	P	R	F1	Acc	P	R	F1
GCNs	98.69	98.91	99.05	98.98	98.13	97.98	98.01	98.00
Bi-GRU+GCNs	99.18	99.51	99.13	99.32	98.56	98.12	98.32	98.22
Bi-GRU+Deep_TNN +Self-attention	99.62	99.67	99.5	99.58	99.2	99.15	98.89	99.01



(a) Binary classification task



(b) Five classification tasks

Figure 8: The ROC curves of malware detection and classification.

4.4.2. Ablation Study of Different Models. The selection of models is important in the experiment. Next, we demonstrate why the HyGNN-Mal model performs well through model ablation experiments. We verify the experimental performance and results of different model combinations, as shown in Table 4. Considering the effectiveness of GCN in dealing with graph structure, we choose it to train the structural information of the source code. We found that performance of Bi-LSTM+GCNs model in Table 3 is similar to the Bi-GRU+GCNs model in Table 4. However, the Bi-GRU takes up less computational resources than Bi-LSTM, so we choose Bi-GRU to process sequence information. As for the problem of long source code sequence information, we use self-attention to effectively solve it. As expected, HyGNN-Mal model performs best compared with other model combination.

To further assess the capabilities of our approach versus the baselines, we compute the receiver operating characteristic (ROC) curve of HyGNN-Mal, shown in Figure 8. For the binary classification task, the AUC value is about 0.9998. In five classification tasks, the AUC values of each category are very close. The AUC value of Adware is the highest at 0.9999, which is close to 1. The aforementioned results show that HyGNN-Mal is effective for Android malware classification task.

4.4.3. The Effect of Source Code Features. In order to evidence the importance of source code analysis in HyGNN-

Table 5: Experimental results of different features comparisons.

Features	Binary-classify				Five-classify			
	Acc	P	R	F1	Acc	P	R	F1
Permissions(P)+APIs	96.38	96.41	96.50	96.45	94.30	94.28	93.98	94.13
Source code ASTs (ASTs)	89.40	89.56	88.79	89.17	86.78	87.01	86.32	86.66
Source code sequence (S)	95.21	95.33	94.20	94.76	91.42	90.60	90.46	90.53
ASTs+S	97.03	97.12	97.14	97.13	96.32	96.20	96.18	96.19
P+APIs+ASTs	97.31	96.95	97.10	97.02	95.01	95.10	94.69	94.89
P+APIs+S	99.01	98.86	98.52	98.69	98.26	98.34	98.19	98.26
P+APIs+ASTs+S	99.62	99.67	99.50	99.58	99.20	99.15	98.89	99.01

Mal, we compare the different feature combinations and observe the performance of them on malware detection. The experimental results are shown in Table 5.

In Table 5, we find that the malware detection effect has been improved, when adding source code semantic information to the P+API combination. It reflects that the analysis of source code level can produce an information gain for the application semantic. Besides, if using only one feature, AST performs the worst, because it only considers the structural information of apps and ignores lexical information. However, different structures may have the same behavior. For example, the code segments of "for" and "while" with different structures both represent loop operations. When we further add the source code lexical information, S, the effect is obviously enhanced, because the combination of sequence and structure information more comprehensively characterizes the applications. Finally, we supplement the behavioral features P+APIs to enhance the semantic information of applications, and it comes with the best result. It is illustrated that the performance of HyGNN-Mal was significantly improved using multiple eigenvectors proposed by us.

5. Conclusion and Discussion

In this paper, we propose a novel automatic detection of Android malware via hybrid graph neural network, HyGNN-Mal. First, we analyze the Android apps from source code level, using AST to represent its structural information, and sequence to represent its syntactic information. Second, we propose the Deep-TNN model to process AST and utilize the self-attention mechanism to process the source code sequences added row and column position information. In addition, we use the Bi-GRU to handle permissions and API features. Finally, a series of experiments are conducted to prove the feasibility and effectiveness of the HyGNN-Mal.

With the increasing of Android malware variants, the challenges of automatic malware detection are as follows: (1) the small datasets, even outdated, lead to an inaccuracy in malware detection. (2) There is a lack of consensus on labeling malwares. (3) The time cost and large storage space of malicious samples are still challenges need to be considered. Depending on our study, we also summarize the future directions.

(i) With the widespread application of DL, it is noteworthy that the combination of DL and source code graph analysis in Android malware detection

(ii) Taking the app descriptions as the purpose comments of used permissions, NLP techniques will be a potential future direction to detect malware

(iii) Given few malicious code samples currently, transferred learning will be a good choice

(iv) Compared with hybrid analysis method, the research on which apps are suitable for static analysis and which are suitable for dynamic analysis is more promising. It will greatly save resources and improve the efficiency of feature extraction

In our next work, we consider using DL technology to process apps suitable for dynamic analysis and NLP technology to process apps suitable for static analysis, for achieving a more accurate classification of families and even striving to effectively detect the newly generated malware.

Data Availability

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] S. G. Stats, "The market share of android smartphones," 2021, http://gs.statcounter.com/osmarket-share/mobile/worldwide.

[2] M. Statista, "Operating systems' market share worldwide from January 2012," https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009.

[3] K. M. Malware Evolution, "The number of malicious installation packages appeared per day," 2021, https://securelist.com/mobile-malware-evolution-2018/89689/.

[4] Mobile Malware Evolution, 2021, https://securelist.com/mobile-malware-evolution-2020/101029/.

[5] T. Kim, B. Kang, M. Rho, S. Sezer, and E. G. Im, "A multimodal deep learning method for android malware detection

using various features," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 773–788, 2019.

[6] M. Fan, J. Liu, X. Luo et al., "Android malware familial classification and representative sample selection via frequent subgraph analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, pp. 1890–1905, 2018.

[7] M. Sewak, S. K. Sahay, and H. Rathore, "An investigation of a deep learning-based malware detection system," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pp. 1–5, Hamburg, Germany, 2018.

[8] T. Song, W. Zheng, P. Song, and Z. Cui, "EEG emotion recognition using dynamical graph convolutional neural networks," *IEEE Transactions on Affective Computing*, vol. 11, no. 3, pp. 532–541, 2020.

[9] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "Cayleynets: graph convolutional neural networks with complex rational spectral filters," *IEEE Transactions on Signal Processing*, vol. 67, no. 1, pp. 97–109, 2019.

[10] Q. Zhang, J. Chang, G. Meng, S. Xu, S. Xiang, and C. Pan, "Learning graph structure via graph convolutional networks," *Pattern Recognition*, vol. 95, pp. 308–318, 2019.

[11] X. Pei, L. Yu, and S. Tian, "AMalNet: a deep learning framework based on graph convolutional networks for malware detection," *Computers & Security*, vol. 93, article 101792, 2020.

[12] H. Safa, M. Nassar, and W. A. R. Al Orabi, "Benchmarking convolutional and recurrent neural networks for malware classification," in *15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pp. 561–566, Tangier, Morocco, 2019.

[13] A. Pektaş and T. Acarman, "Deep learning for effective android malware detection using API call graph embeddings," *Soft Computing*, vol. 24, no. 2, pp. 1027–1043, 2020.

[14] Y. Feng, O. Bastani, R. Martins, I. Dillig, and S. Anand, "Automated synthesis of semantic malware signatures using maximum satisfiability," 2016, https://arxiv.org/abs/1608.06254.

[15] T. Chakraborty, F. Pierazzi, and V. S. Subrahmanian, "Ec2: ensemble clustering and classification for predicting android malware families," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 2, pp. 262–277, 2020.

[16] A. Atzeni, F. Diaz, A. Marcelli, A. Sánchez, G. Squillero, and A. Tonda, "Countering android malware: a scalable semi-supervised approach for family-signature generation," *IEEE Access*, vol. 6, pp. 59540–59556, 2018.

[17] R. Surendran, T. Thomas, and S. Emmanuel, "A TAN based hybrid model for android malware detection," *Journal of Information Security and Applications*, vol. 54, article 102483, 2020.

[18] G. D'Angelo, F. Palmieri, A. Robustelli, and A. Castiglione, "Effective classification of android malware families through dynamic features and neural networks," *Connection Science*, vol. 33, no. 3, pp. 786–801, 2021.

[19] A. F. A. Kadir, N. Stakhanova, and A. A. Ghorbani, "Android botnets: what urls are telling us," *International Conference on Network and System Security. Springer*, vol. 9408, pp. 78–91, 2015.

[20] S. Hou, Y. Ye, Y. Song, and M. Abdulhayoglu, "Make evasion harder: an intelligent android malware detection system," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 5279–5283, Stockholm, Sweden, 2018.

[21] M. Y. Wong and D. I. D. Lie, "A targeted input generator for the dynamic analysis of android malware," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, pp. 21–24, San Diego, CA, USA, 2016.

[22] K. Tam, S. J. Khan, A. Fattori, and L. Cavallaro, "CopperDroid: automatic reconstruction of android malware behaviors," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, pp. 8–11, San Diego, CA, USA, 2015.

[23] W. Niu, R. Cao, X. Zhang, K. Ding, K. Zhang, and T. Li, "OpCode-level function call graph based android malware classification using deep learning," *Sensors*, vol. 20, no. 13, p. 3645, 2020.

[24] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-Droid: deep learning based android malware detection using real devices," *Computers & Security*, vol. 89, article 101663, 2020.

[25] T. Liu, H. Wang, L. Li, G. Bai, Y. Guo, and G. Xu, "Dapanda: detecting aggressive push notifications in android apps," in *34th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE*, pp. 66–78, San Diego, CA, USA, 2019.

[26] A. Guerra-Manzanares, S. Nomm, and H. Bahsi, "In-depth feature selection and ranking for automated detection of mobile malware," *ICISSP*, vol. 1, pp. 274–283, 2019.

[27] J. Gao, L. Li, P. Kong, T. F. Bissyandé, and J. Klein, "Understanding the evolution of android app vulnerabilities," *IEEE Transactions on Reliability (TRel)*, vol. 70, no. 1, pp. 212–230, 2021.

[28] Y. Hu, H. Wang, Y. Zhou et al., "Dating with scambots: understanding the ecosystem of fraudulent dating applications," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, vol. 2019, 2019.

[29] L. Li, T. F. Bissyandé, and J. Klein, "Rebooting research on detecting repackaged android apps: literature review and benchmark," *IEEE Transactions on Software Engineering (TSE)*, vol. 2019, 2019.

[30] K. A. Talha, D. I. Alper, and C. Aydin, "APK auditor: permission-based android malware detection system," *Digital Investigation*, vol. 13, pp. 1–14, 2015.

[31] L. Cen, C. S. Gates, L. Si, and N. Li, "A probabilistic discriminative model for android malware detection with decompiled source code," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 4, pp. 400–412, 2015.

[32] M. Alazab, M. Alazab, A. Shalaginov, A. Mesleh, and A. Awajan, "Intelligent mobile malware detection using permission requests and API calls," *Future Generation Computer Systems*, vol. 107, pp. 509–521, 2020.

[33] J. Su, D. V. Vasconcellos, S. Prasad, D. Sgandurra, Y. Feng, and K. Sakurai, "Lightweight classification of IoT malware based on image recognition," *IEEE 42Nd annual computer software and applications conference (COMPSAC)*, vol. 2, pp. 664–669, 2018.

[34] S. Ni, Q. Qian, and R. Zhang, "Malware identification using visualization images and deep learning," *Computers and Security*, vol. 77, pp. 871–885, 2018.

[35] J. Qiu, J. Zhang, W. Luo, L. Pan, S. Nepal, and Y. Xiang, "A survey of android malware detection with deep neural models," *ACM Computing Surveys (CSUR)*, vol. 53, no. 6, pp. 1–36, 2021.

[36] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, http://arxiv.org/abs/1609.02907.

[37] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is all you need," *Advances in neural information processing systems.*, vol. 30, 2017.

[38] L. Zhang, Z. Huang, W. Liu, Z. Guo, and Z. Zhang, “Weather radar echo prediction method based on convolution neural network and long short-term memory networks for sustainable e-agriculture,” *Journal of Cleaner Production*, vol. 298, article 126776, 2021.

[39] L. Zhang, C. Xu, Y. Gao, Y. Han, X. du, and Z. Tian, “Improved Dota2 lineup recommendation model based on a bidirectional LSTM,” *Tsinghua Science and Technology*, vol. 25, no. 6, pp. 712–720, 2020.

[40] L. Lv, Z. Wu, L. Zhang, B. B. Gupta, and Z. Tian, “An edge-AI based forecasting approach for improving smart microgrid efficiency,” *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2022.

[41] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. E. R. T. Siemens, “Drebin: effective and explainable detection of android malware in your pocket,” *Ndss.*, vol. 14, pp. 23–26, 2014.

[42] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, “AndroZoo: collecting millions of android apps for the research community,” in *13th International Conference on Mining Software Repositories*, pp. 468–471, Austin, TX, USA, 2016.

[43] D. Marcheggiani and I. Titov, “Encoding sentences with graph convolutional networks for semantic role labeling,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1506–1515, Copenhagen, Denmark, 2017.

[44] I. U. Haq, T. A. Khan, and A. Akhunzada, “A dynamic robust DL-based model for android malware detection,” *IEEE Access*, vol. 9, pp. 74510–74521, 2021.

[45] O. Mirzaei, G. Suarez-Tangil, J. M. de Fuentes, J. Tapiador, and G. Stringhini, “Andrensemble: leveraging api ensembles to characterize android malware families,” in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, pp. 307–314, Auckland, New Zealand, 2019.