Hindawi
Mobile Information Systems
Volume 2022, Article ID 4194714, 9 pages
https://doi.org/10.1155/2022/4194714

Research Article

Flow Graph Anomaly Detection Based on Unsupervised Learning

Zhengqiang Yang,[1] Junwei Tian,[2] and Ning Li[3]

[1]*School of Computer Science and Engineering, Xi'an Technological University, Xi'an, China*
[2]*School of Mechatronic Engineering, Xi'an Technological University, Xi'an, China*
[3]*School of Electrical Engineering, Xi'an University of Technology, Xi'an, China*

Correspondence should be addressed to Ning Li; lining83@xaut.edu.cn

Received 5 October 2021; Revised 19 November 2021; Accepted 28 February 2022; Published 31 March 2022

Academic Editor: Yugen Yi

In this paper, a flow graph anomaly detection framework based on unsupervised learning is proposed. Compared with traditional anomaly detection, graph anomaly detection faces some problems. Firstly, the training of a reasonable network embedding is challenging. Secondly, the information data in the real world is often dynamically changing. Thirdly, due to the lack of sufficient training labeled data in most cases, anomaly detection models can only use unsupervised learning methods. In order to resolve these problems, three modules in the framework are proposed in this paper: preprocessor, controller, and optimizer. Additionally, a reasonable negative sampling strategy is applied to generate negative samples to deal with the lack of labeled data. Finally, experiments on real-world data sets are conducted, and the experimental results show that the accuracy of the proposed method reaches 87.6%.

1. Introduction

Anomaly detection has a wide range of application value, e.g., detecting abnormal behaviors in financial transactions, detecting abnormal accounts in social networks, and detecting network attack behaviors. Although there are many traditional and classic anomaly detection methods (density-based methods and distance-based methods), they are not suitable for flow graph data. In recent years, the deep learning method [1] has a positive effect on large-scale graph structure data [2, 3]. It first models the real application scene as a graphical structure and then uses the deep learning model to quickly detect abnormal objects in the data. Although some related solutions [4–11] have been put forward, there are still many problems to be resolved.

Firstly, the dimension of the data object vector is hard to define. Based on the classic random walk model, Perozzi et al. proposed the DeepWalk algorithm [12]. It acquires the path through random walk and treats the path as a sentence embedded in a word. The result can be embedded by training with the word2vec [13] model. The DeepWalk algorithm focuses on depth-first traversal. In contrast, the LINE algorithm [14] focuses on breadth-first traversal. They both are based on the assumption: the more similar the points are, the closer the distance between the vector representations is. After that, the node2vec [15] algorithm is proposed, which makes long-term and considerable improvements on the basis of the DeepWalk algorithm. This algorithm introduces two hyperparameters used as trade-offs to control the balance between depth-first traversal and breadth-first traversal. Compared with the DeepWalk algorithm, the node2vec has stronger presentation ability. The SDNE [16] algorithm is similar to the LINE algorithm, which saves the first-order similarity and the second-order similarity. It pays more attention to the learning of both at the same time. The graph2vec [17] algorithm proposed by Narayanan adopts the embedding method based on the subgraph. Graph neural network [18] is also a very powerful graph embedding method, and it includes graph convolution network and graph attention network [19].

Secondly, data in real life has flexible dynamic characteristics, which makes anomaly detection more difficult. For example, in a social network, two users are associated at a certain time, but the connection may disappear later. This

leads to the change of the network embedded representation vector. The anomaly detection method on static network is no longer suitable for this situation. Recurrent neural network (RNN) [20] can effectively complete research work such as detection, estimation, and prediction. Long short-term memory network [21] is an variant of RNN structure, which includes input gates, output gates, memory cells, and forgetting gates. It can capture potential information effectively in longer time intervals. The Gated Recurrent Unit (GRU) [22] model includes resetting gates and updating gates, which can achieve relatively stable accuracy. The AddGraph model [6] applies gated neural unit based on the attention mechanism for anomaly detection. It adopts neural networks to capture the attributes of the nodes and the potential information of changes.

Thirdly, sufficient labeled data are not easy to obtain in real life. The Auto-Encoder model provides the possibility for unsupervised or self-supervised deep learning. The reconstructed output of the autoencoder is used to compare the original representation of the input. After calculating reconstruction error, data objects with small reconstruction error can be identified as normal objects. Otherwise, they are identified as abnormal objects. The deviation network [7] model separates the encoder module from the autoencoder to form a semi-autoencoder. The semiautoencoder can learn anomaly scores directly from end to end. However, it still needs to obtain abnormal data in advance as training samples.

Variational Auto-Encoder [23] can obtain a probability distribution function that conforms to the data distribution. Inspired by Variational Auto-Encoder, a framework is proposed which uses a novel negative sampling strategy to generate a single or multiple group of abnormal samples based on different percentage of the data. After that, this paper proposes the probability distribution functions to guide the training of normal samples. The contributions of this paper are summarized as follows:

(1) This paper proposes an unsupervised framework for anomaly detection on flow graphs. Unlike the existing unsupervised methods, the proposed framework uses a negative sampling strategy to generate self-defined labels for unsupervised learning, which improves the accuracy rate of anomaly detection

(2) This paper proposes a hierarchical process in the framework, which effectively captures the timing of flow graph and effectively alleviates the impact of unlabeled data for anomaly detection

(3) Extensive experiments are conducted on real-world graph data sets in this paper. The experimental results demonstrate the feasibility and effectiveness of the proposed framework in anomaly detection on unlabeled data

The rest of this paper is organized as follows. In Section 2, the related work is introduced. Section 3 presents the problems to be resolved and describes the proposed method in detail. In Section 4, the performance evaluation shows the superiority of the proposed method. Finally, the conclusion is drawn in Section 5.

2. Related Work

2.1. Network Embedding. The path-based node embedding model comes from the random walk model. The DeepWalk [12] model randomly moves to an adjacent graph node according to a certain transition probability. After that, the adjacent graph nodes repeat the same steps as the second step of the path. After several cycles, a roaming path can be obtained.

The selection of nodes in the path by the DeepWalk model is random, which may ignore many key information. A more effective roaming model node2vec [15] regards walking path as a word-embedded sentence for training. It eliminates the strategy of randomly selecting paths and introduces two hyperparameters: regression parameters and departure parameters. Regression parameters make path selection more inclined to the same order nodes, while deviation parameters make path selection more inclined to the next order nodes. Finally, it weighs the balance between the breadth-first traversal and the depth-first traversal.

The path selection of the NetWalk model [4] is judged based on the degree of the current node. The model is no longer trained based on the idea of word embedding, and it uses a deep autoencoder to reduce and increase the dimensions of the representation vectors of different walking paths in order to learn its potential temporal information. However, whenever the graph structure changes dynamically, the model needs to update all the walking paths comprehensively and retrain the new ones.

The SDNE model [16] is an extension of the LINE model [14], which applies the idea of deep learning to the representation learning of network embedding for the first time. In this model, the autoencoder simultaneously captures the first-order similarity and the second-order similarity of graph nodes. The first-order similarity refers to the local characteristics of the graph nodes, i.e., the correlation between the graph nodes and adjacent nodes. The second-order similarity refers to the global features of the graph nodes, i.e., the correlation with other nonadjacent nodes. By reconstructing the error, the dual features can be learned in an unsupervised manner.

2.2. Timing Capture. Recurrent neural network [20] is a classic model for processing sequential tasks. Its variant, long short-term memory [21], replaces the simple processing functions in the repeated hidden layer modules with gate structures. By adjusting the information flow, this effectively alleviates the problems caused by gradient explosions. The Gated Recurrent Unit network [22] replaces the hidden layer with a variant in a simpler structure. The temporal convolution network model [2] logically connects the three models of causal convolution, dilated convolution, and residual connection into a composite framework. However, it is only suitable for one-dimensional data. In time series data, the importance or priority of data may be higher at one moment and relatively lower at another moment. The self-attention

mechanism network [6] calculates the respective importance weight of each time node according to its priority. Compared with LSTM and GRU, the training time of self-attention mechanism network is significantly reduced.

2.3. Unsupervised Learning. Due to the lack of fully available labeled data in real life, unsupervised learning has become one of the most available modes in anomaly detection. As a classic unsupervised learning method, spectral clustering [24–26] has been widely applied to various computer vision tasks. The traditional autoencoder (AE) [22] can be used to learn phrase representation for sequence data. It is composed of two symmetrical cascade networks: encoder and decoder. They represent the dimensionality reduction and dimensionality increase of the vector, respectively. The product of the dimensionality reduction process is called the intermediate vector, and the product of the dimensionality increase is called the reconstruction result. The Variational Auto-Encoder (VAE) [23] introduces a new function, which can train the probability distribution function of the sample to be trained. The difference between VAE and AE is that the intermediate representation learned by VAE is a function that conforms to the Gaussian distribution. The basic idea of this work is inspired by VAE and AE models.

3. The Proposed Method

3.1. Problem Definition. Flow graph: in a social graph $\mathcal{g}$, let T refer to the time span, and the graph flow in the continuous time can be expressed as $\mathcal{G} = \{\mathcal{g}\}_{t=1}^{T}$. For each graph $\mathcal{g}$, $\mathcal{g} = A^t$ and $A^t \in \mathbb{R}^{n*n}$, where n represents the number of nodes contained in each graph. Each node v refers to each user in the social graph, and each edge $\mathbf{e} = (i, j, w)$ in the adjacency matrix represents the interaction between user i and user j. The notations/symbols used in the following are shown in Table 1.

Abnormal object: abnormal object in flow graph is an object whose behavior is significantly different from that of a normal object, e.g., a node has a very high out-degree at a certain moment, but its out-degree suddenly decreases at the following moment, and such a node is identified as an abnormal node. Note that, if most of the nodes have low out-degrees, some nodes with high out-degrees will also be judged as abnormal nodes.

Problem definition: given a social graph flow $\mathcal{G}$, the goal of this paper is to detect abnormal nodes in $\mathcal{G}$. Specially, the network embedding is learned from the corresponding flow graph.

$$R^{TxNxN} \longrightarrow \text{Network Embedding} \longrightarrow R^{TxNxD}. \quad (1)$$

Then, a certain timing model is used to capture the time correlation of the network embedding and compress it into a temporal network embedding. After that, the unsupervised learning model is used to detect the nodes that show abnormal behavior.

$$R^{TxNxD} \longrightarrow \text{Temporal Processing} \longrightarrow R^{NxD}. \quad (2)$$

3.2. Framework Design. This paper proposes a composite framework for flow graph anomaly detection based on unsupervised learning. The design of the framework is based on the following questions: (1) how to reasonably obtain the network embedding of the data set? (2) How to effectively learn the abnormal behavior in the unlabeled data? The overall structure of this composite frame is shown in Figure 1.

3.2.1. Preprocessor. The Walk Path is adopted to capture the structural characteristics of graph nodes and obtain network embedding. Probability (pre, next) denotes the probability of path selection for random walk method. Two hyperparameters are introduced while making path selection, e.g., return-para and outgoing-para referred as r and o for short, respectively. They are used to control the choice of the roaming path, as shown in formula (3).

$$\text{Probability}(\text{pre}, \text{next}) \begin{cases} \frac{1}{\text{r}}, \text{if next} = \text{pre} \\ 1, \text{if next near pre} \\ \frac{1}{\text{o}}, \text{if next near now} \end{cases} \quad (3)$$

As shown in Figure 2, the blue node refers to the pre node, and the gray node refers to the now node. The colorless nodes refer to the possible selection of the next node. Therefore, the three equations represent the possibility that the next step will return to the pre node. The next step will jump to the 1-hop neighbor of the pre node, and the next step will outgo to the 2-hop neighbor of the pre node. According to formula (4), the edge weight information can also be taken into account in the graph structure.

$$\text{Probability} = \text{Probability}(\text{pre}, \text{next}) * \text{weight}(\text{now}, \text{next}). \quad (4)$$

Due to the lack of prior knowledge, while dividing the samples to be tested into normal samples, a negative sampling strategy is adopted to generate abnormal samples, as shown in formula (5).

$$\text{SamplingPro}(i) = \frac{\text{Degree}_i}{\text{Degree}_i + \text{Degree}_j}. \quad (5)$$

Given an edge $\mathbf{e} = (i, j)$: for node i, there is a probability of $\text{Degree}_i/\text{Degree}_i + \text{Degree}_j$ to replace it with its neighbor nodes; for node j, there is a probability of $\text{Degree}_j/\text{Degree}_i + \text{Degree}_j$. In this way, edges that do not exist in the original graph flow are generated. If the new edge already exists in the original graph, it needs to be resampled until the edge does not exist in the original graph.

SEModel is adopted to calculate the weight of the graph data and capture its time correlation at each moment in the flow graph. First, for a given matrix $X \in 50 * 1899 * 128$, parameter normalization is used to add

Table 1: The notations/symbols.

Notations/symbols	Meaning
T	The time span of the data to be trained
g	The graph of the formalized transformation of interactive information within a certain time range
$\mathcal{G}$	The graph flow in the continuous time
n	The number of nodes
e	Edge
w	Weight value of the given edge
A	Adjacency matrix
R	Real number set
N	The number of nodes
r	Return-para
o	Outgoing-para
D	The number of dimension
X	The positive sample
X'	The negative sample

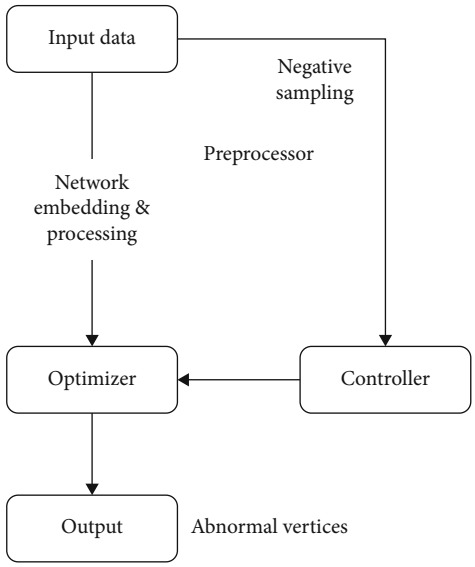

Figure 1: Overall structure of the framework.

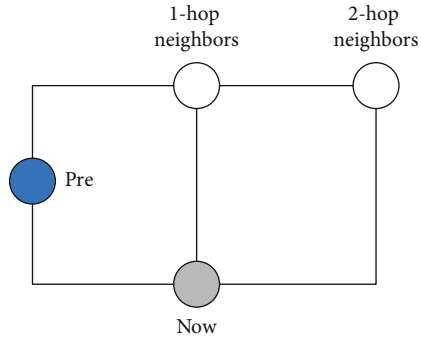

Figure 2: Schematic diagram of random walk algorithm strategy.

them to a 50×1 matrix.

$$G = \text{Fuse}(G). \tag{6}$$

Next, with the help of MLP (multilayer perceptron) and Softmax function, it is normalized to a 1×50 -dimensional weight matrix to represent the weight of each time node, as shown in formula (7).

$$\text{Weight} = \text{SoftMax}(\text{MLP}(G)). \tag{7}$$

Finally, substitute the weight matrix into the original data to obtain comprehensive graph of time sequence, as shown in formula (8).

$$G_{\text{final}} = \text{Weight} * G. \tag{8}$$

3.2.2. Controller. Variational Auto-Encoder is used to train the probability distribution function of negatively sampled data. The main idea is shown in Figure 3.

The probability distribution function conforming to the normal distribution is trained from the samples. After that, sampling is carried out to continuously generate new samples to guide the training of the optimizer. The operation of encoder is similar to that of the self-encoder, but the difference is that the intermediate representation is not the product of feature extraction and a hidden layer conforming to the normal distribution.

$$(\mu_t, \sigma_t) = \text{Variational AutoEncoder}\left(X_{\text{anomaly samples}-t}\right). \tag{9}$$

The controller is equivalent to adding strong constraints to the training of the optimizer. It can effectively prevent overfitting and underfitting.

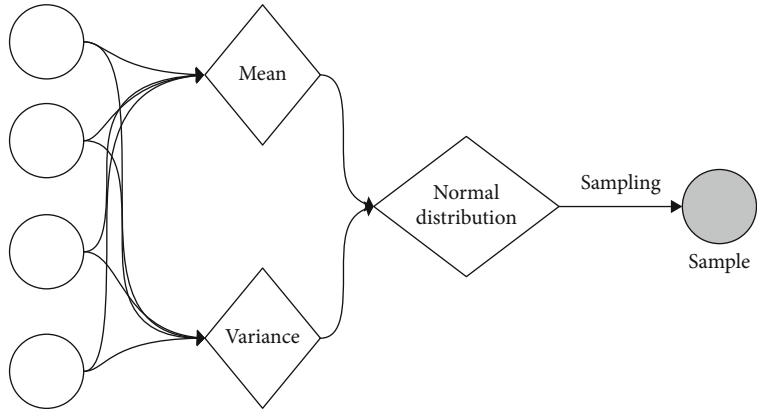

Figure 3: The basic idea of controller.

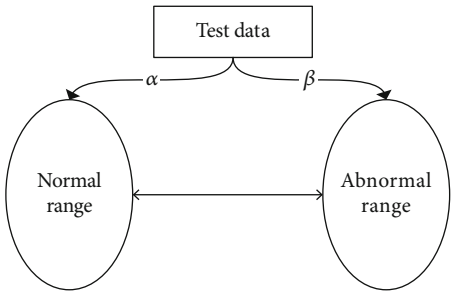

Figure 4: The idea of objective function.

3.2.3. Optimizer. Autoencoder is used to reduce and increase the dimension of the representation vector to reasonably extract valuable features, reconstruct similar output results, and finally distinguish normal samples from abnormal samples. Firstly, the reconstructed output is obtained by reducing dimension and increasing dimension.

$$X_{\text{reconst}} = \text{Decoder}(\text{Encoder}(X)) \tag{10}$$

Secondly, the following formula (11) is used to make the gap between the negative sample and the reconstruction result as large as possible and the gap between the positive sample and the reconstruction result as small as possible.

$$\text{Loss}_{\text{Train}} = p * \text{Distance}(X, X_{\text{reconst}}) - q * \text{Distance}\left(X', X'_{\text{reconst}}\right), \tag{11}$$

where X represents the positive samples, and X' represents the negative samples. Through optimized training, the normal samples can be as close as possible to the reconstructed samples, and the abnormal samples can be as far away as possible from the reconstructed samples. After training, the model is saved for the subsequent test experiments. For each sample imported into the optimizer, a negative sample is randomly taken from the controller to reconstruct the difference, as shown in formula (12).

$$\text{Distance}_{\text{Test}} = \alpha\,\text{Distance}A(X_{\text{reconst}}, X) + \beta\text{Distance}B\left(X, X_{\text{sample}}\right), \tag{12}$$

where α and β control the distance of two differences, respectively. Distanc e$A()$ controls the difference between the reconstruction result and the sampled samples. The smaller the difference is, the closer it is to the anomaly. Distance $B()$ controls the difference between the reconstruction result and the input samples. The greater the difference is, the closer it is to the anomaly. As shown in Figure 4, Distance A () and Distance B () drag the test samples back and forth between normal and abnormal samples. If it is closer to the abnormal samples, Distance A () is greater, and Distance B () is smaller. Under normal circumstances, the larger the distance is, the closer it is to the abnormal samples.

3.3. Pseudocode. Algorithm 1 clarifies the design ideas of the proposed method in this paper. The core idea is to use the controller trained by negative sampling to guide anomaly detection. The pseudocode is composed of three main functions, corresponding to the three modules of the preprocessor, the controller, and the optimizer, respectively. In the function PREPROCESSOR(), NODE2VEC() is adopted to capture the structural characteristics of graph nodes and obtain network embedding. A self-attention mechanism SEMODEL() is adopted to calculate the weight of the graph data at each moment in the flow graph and to capture the time correlation. In the function CONTROLLER(), Variational Auto-Encoder is used to train the probability distribution function of negatively sampled data. In the function OPTIMIZER(), AUTO-ENCODER() is used to reduce and increase the dimension of the representation vector to reasonably extract valuable features. And then, reconstruct similar output results and distinguish normal samples from abnormal samples. Through function OPTIMIZER(), the normal samples can be as close as possible to the reconstructed samples, and the abnormal samples can be as far away as possible from the reconstructed samples.

```
Input: Graph stream G
Output:abnormal object
1:function PREPROCESSOR(G)
2:                  if called by Controller then
3:            3 types of Anomaly Samples A←NEGATIVESAMPLING(G)
4:            A←NODE2VEC(A)
5:            return A
6               else
7:              X←NODE2VEC(G)
8:              X←SEMODEL(X)
9:              return X
10:              end if
1l:end function
12:
13:function CONTROLLER(G)
14:          A ← PREPROCESSOR(G)
15:          for a in A do
16:                  Gaussian distribution←VARIATIONAL AUTO-ENCODER(a)
17:                  Save Model Gaussian distribution
18:          end for
19:end function
20:
2l:function OPTIMIZER(X-test)
22:          for x in X-test do
23:          xReconst←AUTO-ENCODER(x)
24:          xSample←GAUSSIAN DISTRIBUTION
25:          Deviation←LOSSFUNCTION(x,xReconst, xSample)
26:          end for
27:end function
```

Algorithm 1: Flow graph anomaly detection.

Table 2: Overview of the data set.

Dataset	Vertex	Max degree	Avg degree	Time stamp
UCI message	1899	255	14.57	59

Table 3: Confusion matrix for anomaly detection result.

	1	0
1- > positive (P)	True positive (TP)	False positive (FP)
0- > negative (N)	False negative (FN)	True negative (TN)

Table 4: Accuracy fluctuation range.

Ablation experiment	Fluctuation range
Scheme 1	0.691 ⟶ 0.788
Scheme 2	0.770 ⟶ 0.790
Scheme 3	0.847 ⟶ 0.865
Scheme 4	0.856 ⟶ 0.877
Scheme 5	0.875 ⟶ 0.890

4. Experimental Results and Analysis

4.1. Datasets. The experiments in this paper are evaluated on the UCI message data set [27]. UCI message is a simple data set extracted from the interaction between users at certain moments in an online social network of the University of California, Irvine. In this data set, each node represents an account in the social network, and each edge represents the interaction between two accounts. As shown in Table 2, the data set contains a total of 1899 nodes, the maximum node degree is 255, and the average node degree is 14.57. The time is divided into 59 intervals, and each interval contains a total of 1000 edges.

4.2. Network Embedding. The experiments are performed in PyCharm and Jupyter platforms. The initial data set is preprocessed first by preprocessor, and the graph format data are obtained. Random walk is used to obtain walking path of each node. And then, the path is imported into word2vec model to obtain the network embedding. After the training, the network embedding representation of all graphs is obtained. There are 59 original flow graphs. The first 50 are used as the training set and the last 9 as the test set. The 50 graphs are divided into 5 groups, and SEModel is conducted 5 times, respectively. Finally, the flow graph data are compressed into one graph.

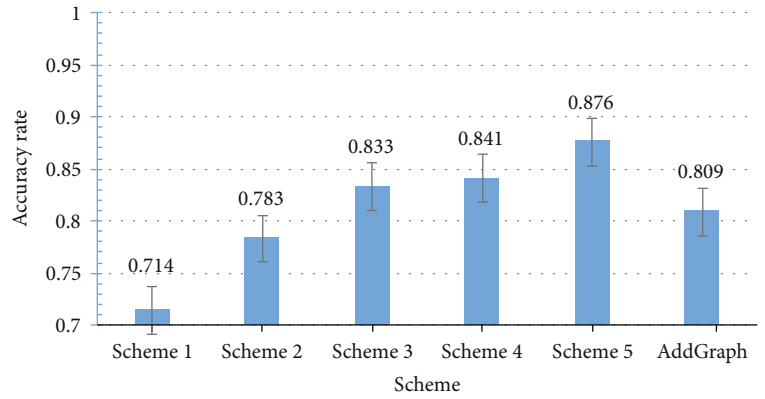

Figure 5: Accuracy comparison.

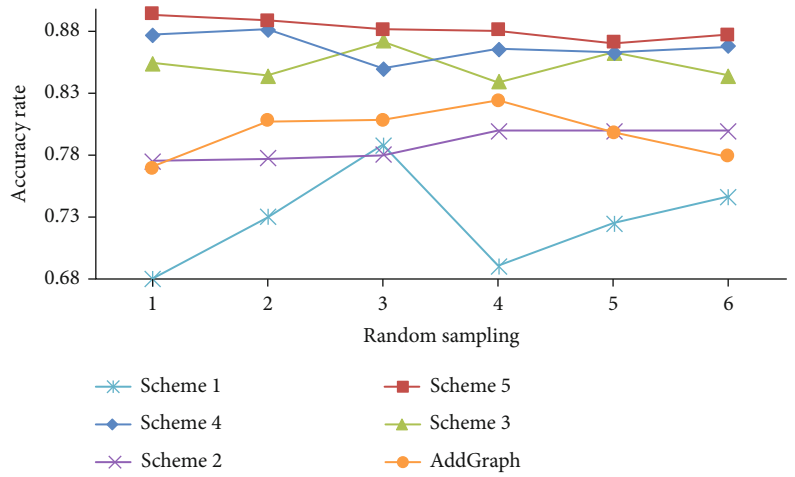

Figure 6: Stability of accuracy rate comparison.

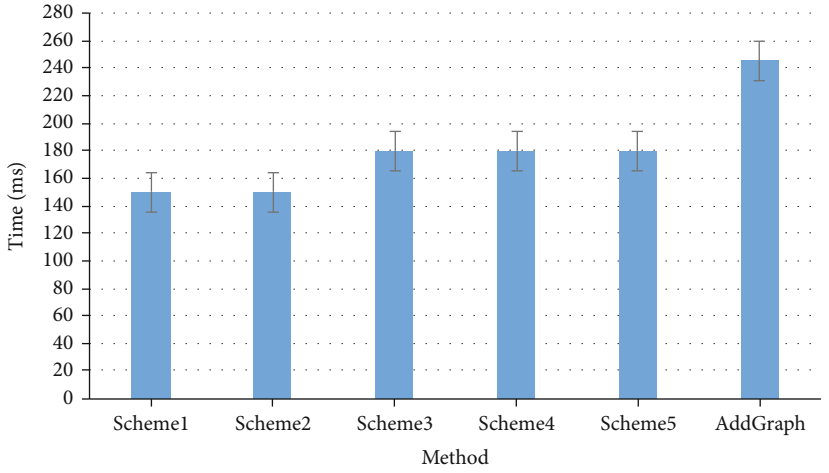

Figure 7: Time-consuming of each scheme.

4.3. Ablation Experiment. This section introduces the design ideas of the ablation experiment and analyzes the respective effects of each module. Five different schemes are used for comparison: scheme 1 (without SEModel, no controller), scheme 2 (with SEModel, no controller), scheme 3 (controller using 5% negative sampling data), scheme 4 (controller using 5% +10% +15% negative sampling data), and scheme 5 (controller using 5% +10% negative sampling data). We also compared the proposed method with the state-of-the-art method called AddGraph [6].

For each ablation experiment, multiple random trials are performed. There are a total of 59 flow graphs in the original

data. To begin with, the first 50 graphs are taken as training data from which a few random graphs are negatively sampled and turned into abnormal graphs. When testing, input all normal graphs and negative sampling graphs in the form of flow graph. Secondly, 200 abnormal nodes are sampled, and the 200 nodes with the largest final distance are maintained in the following test. Then, calculate the same number of nodes as 200 abnormal nodes. Finally, perform matching and count TP (true positive), FP (false positive), FN false negative), and TN (true negative) based on the experimental data. As shown in Table 3, TP represents the number of abnormal nodes that are correctly marked by the proposed method, FP means that it is actually a normal node but it is marked as an abnormal node, FN represents the number of normal nodes marked as abnormal nodes, and TN represents the number of normal nodes that are marked correctly by the proposed method. The accurate rate is measured as $(TP + TN)/(TP + FP + FN + TN)$. Table 4 lists the accuracy fluctuation values of the experiment.

The accuracy and stability comparison test results of various schemes are shown in Figures 5 and 6, respectively. Scheme 1 using only node2vec has the lowest accuracy. Scheme 3, scheme 4, and scheme 5 all contain controller, and their accuracy is significantly higher than schemes without controller (scheme 1 and scheme 2). It verifies the necessity of the controller. Scheme 4 uses three sets of sampling samples. The effect of the controller is higher than that of using only 5% of the sample data, but lower than scheme 5. This is because when the sample data is too small, the result will show a state of underfitting. When the sample data is too much, the negative sample data is too much, and the accuracy rate is reduced. AddGraph [6] is a state-of-the-art method, which adopts GCN and GRU to process anomaly detection in a flow graph. The accuracy rate of AddGraph is between scheme 1 (use Node2Vec) and the proposed method (scheme 3, scheme 4, and scheme 5). Six different random sampling are performed to measure the stability of accuracy rate of the related methods. For different random sampling, due to the lack of sequential capture in scheme 1, its accuracy rate has a large fluctuation value. Scheme 2, scheme 3, scheme 4, and scheme 5 adopt SEModel to capture timing, and they outperform scheme 1 in terms of stability. AddGraph extends the original GCN to support temporal information using GRU, from the perspective of the model, and AddGraph is a supervised method. Due to the supervised method is not suitable to perform anomaly detection when the explicit labelled data is insufficient, AddGraph employs a selective negative sampling and margin loss in model training to alleviate this problem. Then, AddGraph can also be considered as a semisupervised fashion on the whole. Although with the aid of selective negative sampling and margin loss in model training, the accuracy and stability cannot be guaranteed when the explicit labeled data is insufficient for AddGraph. For unsupervised method, the accuracy of anomaly detection highly depends on which train data they used. Since the feature of degree change is obtained from the training dataset, the accuracy of anomaly detection can only be guaranteed when the distribution of testing dataset is same with the training dataset.

Random walk is used to obtain the walk path of nodes in the network. The detail of the process has been described in Section 3. After obtaining the walking path of the network, import the path into the node2vec model to obtain the corresponding network embedding. The time consumption of each scheme is shown in Figure 7. Compared with scheme 1 and scheme 2 (without controller), the time consumption of scheme 3, scheme 4 and scheme 5 is not obvious, but their accuracy is significantly higher than scheme 1 and scheme 2. Compared with the proposed method, AddGraph has to train GCN and GRU. And its time-consuming depends on the convergence speed of training process. In the experiments, its time-consuming is more than the proposed unsupervised method.

5. Conclusions

This paper proposes a composite framework for flow graph anomaly detection based on unsupervised learning. The proposed framework applies the controller for unsupervised learning. SEModel based on self-attention mechanism is adopted to effectively capture the timing features of the flow graph. In addition, a reasonable negative sampling strategy is proposed to solve the problem of insufficient labeled data. Finally, an experimental comparison on a real data set is conducted to verify the superiority of the proposed method.

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 with any financial organizations regarding the material reported in this paper.

Acknowledgments

This work was supported in part by Shaanxi Key Research and Development Plan (2021GY-318, 2022GY-182), National Natural Science Foundation of China (52177193), and China Scholarship Council (CSC) State Scholarship Fund International Clean Energy Talent Project (Grant Nos. [2018]5046, [2019]157).

References

[1] L. Yann, B. Yoshua, and H. Geoffrey, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] H. Li, J. Huang, H. Yuan et al., "A two-phase method to balance the result of distributed graph repartitioning," *IEEE Transactions on Big Data*, p. 1, 2021.

[3] H. Li, H. Yuan, J. Huang, X. Ma, J. Cui, and J. Yoo, "Edge repartitioning via structure-aware group migration," *IEEE Transactions on Computational Social Systems*, pp. 1–10, 2021.

[4] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang, "NetWalk:a flexible deep embedding approach for anomaly detection in dynamic networks," in *24th ACM*

SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 2672–2681, London, England, August 2018.

[5] M. Yoon, B. Hooi, K. Shin, and C. Faloutsos, "Fast and accurate anomaly detection in dynamic graphs with a two-pronged approach," in *25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 647–657, Anchorage, AK, August 2019.

[6] L. Zheng, Z. Li, J. Li, Z. Li, and J. Gao, "AddGraph: anomaly detection in dynamic graph using attention-based temporal GCN," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 4419–4425, Macao, China, August 2019.

[7] G. Pang, "Deep anomaly detection with deviation networks," in *25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 353–362, Anchorage,America, August 2019.

[8] Z. Wu, "Graph WaveNet for deep spatial-temporal graph modeling," 2019, http://arxiv.org/abs/1906.00121.

[9] D. Eswaran, "SpotLight: detecting anomalies in streaming graphs," in *24th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1378–1386, London, ENGLAND, August,2018.

[10] L. Shen, J. Chen, S. He, E. Xu, H. Lv, and Z. Zhou, "Intelligent fault diagnosis under small sample size conditions via Bidirectional InfoMax GAN with unsupervised representation learning," *Knowledge-Based Systems*, vol. 232, article 107488, 2021.

[11] L. Jie, L. Meiqi, W. Xie, L. Yunsong, and J. Xiuping, "Spectral mapping with adversarial learning for unsupervised hyperspectral change detection," *Neurocomputing*, vol. 465, pp. 71–83, 2021.

[12] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: online learning of social representations," in *20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701–710, New York, NY, USA, August 2014.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, http://arxiv.org/abs/1301.3781.

[14] J. Tang, Q. Meng, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: large-scale information network embedding," in *24th International Conference on World Wide Web*, pp. 1067–1077, Florence, ITALY, March 2015.

[15] G. Aditya and L. Jure, "node2vec: scalable feature learning for networks," in *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855–864, San Francisco, CA, August 2016.

[16] D. X. Wang, "Structural deep network embedding," in *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1225–1234, San Francisco,AMERICA, August 2016.

[17] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "graph2vec: Learning Distributed Representations of Graphs," 2017, http://arxiv.org/abs/1707.05005.

[18] 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.

[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," 2018, http://arxiv.org/abs/1710.10903.

[20] J. Chung, V. Gulcehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks," in *32nd International Conference on Machine Learning*, pp. 2067–2075, Lille, FRANCE, July 2015.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] K. Cho, B. van Merrienboer, C. Gulcehre et al., "Learning Phrase Representations Using RNN Encoder Decoder for Statistical Machine Translation," 2014, http://arxiv.org/abs/1406.1078.

[23] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," 2014, http://arxiv.org/abs/1312.6114.

[24] Z. Li, F. Nie, X. Chang, Y. Yang, C. Zhang, and N. Sebe, "Dynamic affinity graph construction for spectral clustering using multiple features," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 12, pp. 6323–6332, 2018.

[25] Z. Li, F. Nie, X. Chang, L. Nie, H. Zhang, and Y. Yang, "Rank-constrained spectral clustering with flexible embedding," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 12, pp. 6073–6082, 2018.

[26] Z. Li, L. Yao, X. Chang, K. Zhan, J. Sun, and H. Zhang, "Zero-shot event detection via event-adaptive concept relevance mining," *Pattern Recognition*, vol. 88, pp. 595–603, 2019.

[27] R. A. Rossi and N. K. Ahmed, "The Network Data Repository with Interactive Graph Analytics and Visualization," in *AAAI*, Austin, Texas, 2015.