Electronic Engineering and Informatics
G. Izat Rashed (Ed.)

doi:10.3233/ATDE240086

An Anti-Money Laundering Method Based on Spatio-Temporal Graph Convolutional Networks

Pengxiang NING, Pengwei WANG[1], Zhaohui ZHANG
School of Computer Science and Technology, Donghua University, Shanghai, 201620, China

Abstract. Anti-money laundering is crucial for maintaining financial security and social stability. In this paper, we propose an anti-money laundering method based on spatio-temporal graph convolution AT-GCN. AT-GCN consists of GCN adaptive parameter update method, oversampling method across time steps, and similarity-based weighted aggregation method. The effectiveness of AT-GCN is demonstrated using a real dataset of Bitcoin money laundering transactions.

Keywords. Anti-money Laundering; Dynamic graph; GCN

1. Introduction

The United Nations Office on Drugs and Crime estimates that between $500 billion and $1 trillion is laundered globally every year [1], with a substantial portion of these illicit funds originating from drug trafficking, robbery, and terrorist activities. In the present day, the proliferation of money laundering activities poses a grave risk not only to financial security but also to social stability.

The rule-based approach is frequently harnessed in Anti-Money Laundering (AML) strategies, attributed to its simplicity and interpretability. However, these approaches often rely on extensive prior knowledge from domain experts, and integrating new rules with existing ones can lead to performance issues and high iteration costs for updates [2]. Currently, more and more AML systems are using machine learning techniques [3].

Traditional machine learning cannot exploit the rich interactions in money laundering transactions, so GNNs [4-6] have been used in recent years for anti-money laundering and transaction fraud. But some problems still exist. A primary issue is an inherent imbalance in labeling, as money laundering activities only comprise a minuscule fraction of ordinary transaction activities. This label imbalance can critically impair the performance of the model. Additionally, the graph structure in real financial transaction networks is changing all the time. Although GNN has achieved great success in static graphs [7, 8], they have not made good progress in the study of dynamic graphs. To address these problems, this paper proposes an anti-money laundering model based on spatio-temporal graph convolutional networks (AT-GCN).

[1] Corresponding author: Pengwei WANG, School of Computer Science and Technology, Donghua University, e-mail: wangpengwei@dhu.edu.cn

In this paper, a new model AT-GCN is proposed to detect suspicious money laundering transactions, and an oversampling method across time steps is proposed to solve the label imbalance problem. An aggregation method based on similarity metric is proposed to mitigate the negative impact of noisy data.

2. Methodology

2.1. Model overview

The model consists of three main parts: the LSTM dynamic update module, the time window oversampling, and the similarity-based aggregation module. Figure 1 shows the framework diagram of the model.

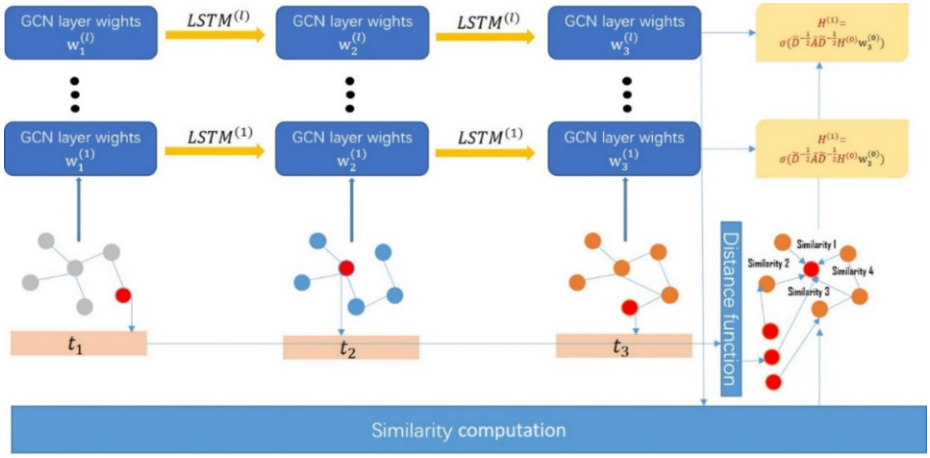

Figure 1. AT-GCN model architecture with time step set to 3 as an example

2.2. GCN adaptive parameter update method

In the process of money laundering detection, since graphs are spatially structured, the use of GCNs for learning the spatial structure of dynamic graphs is considered. Considering the temporal relationship of dynamic graphs, the GCN weights at each moment are also temporally correlated when using GCN for learning dynamic graphs [9]. In the dynamic graph, the method implements an adaptive parameter update mechanism for the GCN using the LSTM [10]. This update mechanism makes the weight parameters of the convolution change when the graph structure changes. The model takes as input to the LSTM the weight parameter W_{t-1}^{l} at moment $t-1$, l-th layer of the graph neural network, and updates the output W_t^l.

In the method, the weight matrix $W_t^{(l)}$ of the l-th layer of the GCN at time t is used as the input of the LSTM. Finally, the obtained weight parameters are passed into the GCN so as to realize the adaptive parameter update of the GCN. Defined as Eq. (1).

$$W_t^l = LSTM(W_t^l) \tag{1}$$

2.3. Oversampling methods across time steps

Since the dynamic graph snapshots in each time step suffer from label imbalance, it is important to balance the labels. However, dynamic graphs are often time-dependent, so adding the fraudulent nodes in the previous time step to the dynamic graph snapshots in the later time step will have a positive effect. The method establishes connections between nodes in different time steps by calculating the distance between the fraudulent nodes in the previous time window and all nodes in the later time step.

Oversampling first requires that the fraudulent nodes in each time step within the time window be selected, and the sampling equations are given in Eq. (2) and Eq. (3):

$$Fsample = \sum_{t}^{t+Tw} Sample_t \tag{2}$$

$$N = N.add(Fsample) \tag{3}$$

where $Sample_t = \{v | y_v = fraud, v \in \mathcal{G}_t\}$, Tw denotes the time window size. By changing different time windows, the time steps to be sampled and the number of samples can be controlled.

As the fraudulent nodes from the previous time window are added to the graph of the later time step, it is necessary to consider how to connect these fraudulent nodes to the nodes of the later time step. The widely used distance function is the Euclidean distance function, defined as Eq. (4).

$$D = ||x_v - x_u|| \tag{4}$$

where $x_v \in R^d$ denotes the features of node v. The Euclidean distance function is used directly here in order to link nodes of different time steps by this metric.

After calculating the distances, the sampled nodes need to be added to the current time step, and links need to be established with the nodes in the current time step, as shown in Eq. (5) and Eq. (6). By calculating the difference between the Euclidean distance of each node in the current time step and the Euclidean distance of the sampled nodes, the connection is established with the sampled nodes that are close to each other. It allows nodes with different time steps to also participate in the training and have a positive impact on the training of the current time step.

$$Top = Select(\{Sort(D_v - D_u) | u \in Fsample\}) \tag{5}$$

$$E = E.add(v, u) \quad where\ u \in Top, v \in V \tag{6}$$

where the $Select()$ function represents the selection of the sampling node u with the smallest distance difference from node v. Eq. (6) represents the establishment of the connection relationship between node v and node u.

2.4. Aggregation method based on similarity weighting

Considering that GCN uses an average aggregation approach in the aggregation process, it does not distinguish the importance between neighbors. When the camouflages associate themselves with a larger number of benign individuals, using this approach

may not be effective in identifying the camouflage, so a similarity-based metric is designed here. Thus, inspired by previous work, a parametric distance function that combines embedded and real label information is used here, defined as Eq. (7).

$$\mathcal{D}is^{(l)}(v,u) = ||\mathcal{D}^{(l)}(h_v^{(l)}) - \mathcal{D}^{(l)}(h_u^{(l)}))|| \tag{7}$$

where $\mathcal{D}^{(l)}(h_v^{(l)})$=$MLP^{(l)}(h_v^{(l)})$ denotes the probability of predicting fraud based on node embedding, and the distance function is defined as the difference between the predicted probability of node v and node u.

In previous work, less work has been done in the graph neural network aggregation process, and the methods tend to be averaging, taking the maximum value, etc. These methods do not take into account the importance of different neighbors for the nodes. Therefore, in order to preserve the relational importance, the weighting is done directly using the similarity, and the transformation is done by the adjacency matrix, which is converted into the weight matrix of the central node and the neighbor nodes, so that the aggregation operation can be done while saving the computational cost.

$$H_t^{(l+1)} = \sigma(Z^{-\frac{1}{2}}AjZ^{-\frac{1}{2}}H_t^{(l)}LSTM\{W_t^{(l)}\}) \tag{8}$$

Eq. (8) shows the GCN using the similarity aggregation method, where Z is the sum of the distances between the central node and its neighbor nodes, i.e., $Z = SUM(\mathcal{D}is^{(l)}(v,u))$, and Aj is the similarity weighting matrix of the dynamic graph at moment t, $Aj = I + Dis(v,u)$.

The loss function can be defined as a cross-entropy loss function, as shown in Eq/ (9).

$$\mathcal{L}_{GNN} = -\sum_{v\in V}[\alpha_1 y_v log p_v + \alpha_2(1-y_v)\log(1-p_v)] \tag{9}$$

where $p_v = MLP(h_v^{(L)})$ and y_v denotes the label of node v.

3. Experiment and Analysis

3.1. Comparison models

- *GCN* [4]: The core idea of GCN is to convolve the features of a node with the features of its neighbor nodes, so as to carry out information propagation and feature learning on the graph.
- *GAT* [5]: By introducing an attention mechanism, GAT is able to adaptively learn the importance weights between nodes for information propagation and feature learning on the graph. GAT is also trained using snapshot graphs at each time step and does not consider the dynamics of the graph.
- *EvolveGCN-O* [9]: EvolveGCN is an adaptive graph convolutional network model along the time dimension. EvolveGCN uses RNN to evolve GCN parameters to capture the dynamics of the graph sequence. EvolveGCN-O is a version that uses LSTM for dynamic modeling.

- *EvolveGCN-H* [9]: EvolveGCN-H is the version of EvolveGCN that uses GRU for dynamic modeling.

3.2. Experimental setting

AT-GCN was implemented based on Pytorch 1.10.2, and each dataset was divided into three parts, where the data in the first 31 time steps served as the training set, the data in the 32-36 time steps served as the validation set, and the data in the 37-49 time steps served as the test set. All experiments were run on Python 3.8.5, Ubuntu 18.04.1, GeForce RTX 2080 Ti GPU, 16GB RAM, Intel(R) Xeon(R) Silver 4214 CPU @ 2.20 GHz. For EvolveGCN and its variants directly using the source code provided by the authors, GCN, the GAT is implemented based on DGL [4].

3.3. Results

This paper uses the Elliptic [3] dataset for model validation, which is currently the largest dataset of Bitcoin transactions in the world. Because the first 94 features in the dataset are local information of transactions and the last 72 features are aggregated, the experiments compare the effect of using the first 94 features with the effect of using all the features separately to ensure the fairness of the experiments.

3.3.1. Performance comparison.

Table 1 and Table 2 show the performance of AT-GCN and other baseline algorithms on the Elliptic dataset, using the first 94 local features and all features experimented in this paper. It can be seen that the majority of the metrics outperform the other baseline algorithms.

Table 1. Model effect using 94 local features

Method	F1	Recall	Precision
AT-GCN	**0.5786**	0.6083	**0.5517**
EvolveGCN-O	0.5095	**0.6348**	0.4255
EvolveGCN-H	0.3418	0.5253	0.2533
GAT	0.4028	0.3641	0.4508
GCN	0.3848	0.3744	0.3959

Table 2. Model effect using all features

Method	F1	Recall	Precision
AT-GCN	**0.6143**	**0.5311**	**0.7283**
EvolveGCN-O	0.5384	0.4689	0.6320
EvolveGCN-H	0.2181	0.3422	0.1600
GAT	0.4710	0.4159	0.5429
GCN	0.5316	0.4747	0.6041

3.3.2. Ablation experiment.

To verify the effectiveness of the proposed method, ablation experiments were conducted using the first 94 local features and all features for different methods, respectively. The experimental results show that AT-GCN is superior to GCN and the single method,

proving the effectiveness of the combination of the three methods. *GCN+W* is a simplified version of the AT-GCN proposed in this paper, using only the LSTM for adaptive parameter updating. *GCN+O* is a simplified version of the AT-GCN, which uses only the oversampling method across time steps. *GCN+A* is a simplified version of AT-GCN using only the similarity-based weighted aggregation method.

In the case of using only 94 local features, Precision increased by 14.2%, Recall increased by 22.6%, and F1 increased by 17.9% after using the GCN adaptive parameter update method proposed in this paper. In the case of using only oversampling methods across time steps, the effect of GCN is also increased, with Precision improving by 10%, Recall by 20.6%, and F1 by 15%. In the case of using only the similarity-based weighted aggregation method, Precision increased by 11.4%, Recall by 13.9%, and F1 by 12.7%. The experiment is shown in Figure 2.

In the case of using all the features, the addition of more information leads to an improvement in the effectiveness of all the models, in which case using one of the spatio-temporal graph convolutional laundering methods alone still improves the results much more than the GCN. After using the GCN adaptive parameter update method proposed in this paper, Precision increased by 13.5%, Recall increased by 2%, and F1 increased by 6.2%. In the case of using only the inter-time-step sampling method, the effect of GCN is also improved, with Precision improving by 3.7% and F1 improving by 0.8%. In the case of using only the similarity-based weighted aggregation method, Precision increased by 3.5%, Recall increased by 0.6%, and F1 increased by 1.7%. The experiment is shown in Figure 3.

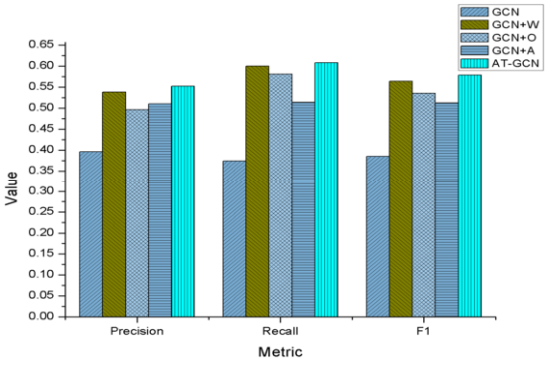

Figure 2. Performance comparison using 94 local features

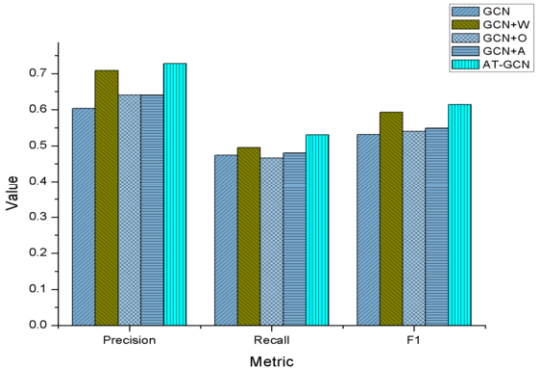

Figure 3. Performance comparison using all features

3.3.3. Sensitivity analysis.

To verify the robustness of AT-GCN, its performance was compared on the Elliptic dataset with different experimental parameters and parameter sensitivity experiments were conducted. For presentation purposes, only 94 local features are used for the experiments. Figure 4 and Figure 5 give the results of parameter sensitivity under several different parameters.

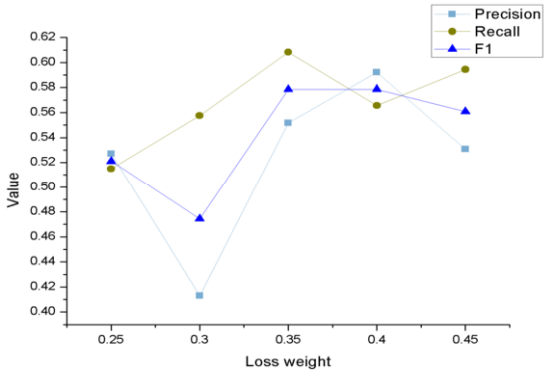

Figure 4. The impact of different loss function weights on model performance

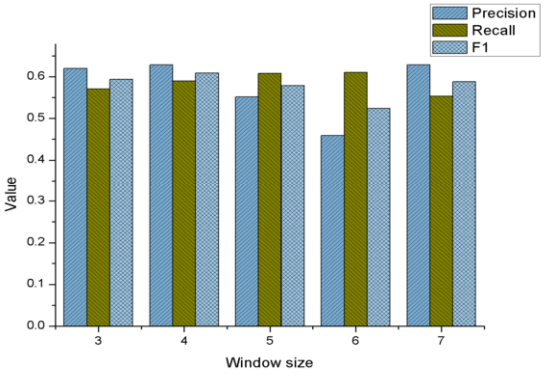

Figure 5. The impact of different time window sizes on model performanc

4. Conclusion

This paper proposes an anti-money laundering strategy based on spatio-temporal graph convolution. The strategy is divided into three parts, which are GCN adaptive parameter update, oversampling across time steps, and weighted aggregation based on similarity. The GCN adaptive parameter update method uses LSTM to learn the trainable weight parameters and obtain the temporal correlation of the weight parameters, so that the model parameters can be updated adaptively with the change of the graph structure. In oversampling across time steps, the fraud nodes within the time window are oversampled. By calculating the distance between these oversampled nodes and the nodes in the current time step, the relationship between the oversampled nodes and the nodes in the current time step is established, and the detection effect of the model on fraud nodes is increased.

Finally, in the aggregation method of GCN, the label correlation between nodes and neighbors is not considered in some previous works. Therefore, the similarity between nodes is calculated by MLP and the similarity is treated as a weighted value, which can increase the effectiveness of the model. The final test results show that AT-GCN has reached the current optimal in most indicators.

Acknowledgment

Pengwei Wang is the corresponding author. This work was partially supported by the National Natural Science Foundation of China (NSFC) under Grant 61602109, DHU Distinguished Young Professor Program under Grant LZB2019003, Shanghai Science and Technology Innovation Action Plan under Grant 22511100700, Fundamental Research Funds for the Central Universities, the Key Innovation Group of Digital Humanities Resource and Research of Shanghai Municipal Education Commission.

References

[1] R. W. Baker, "The biggest loophole in the free-market system," Washington Quarterly, vol. 22, pp. 29-46, 1999.
[2] Z. Chen, L. D. Van Khoa, E. N. Teoh, A. Nazir, E. K. Karuppiah, and K. S. Lam, "Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection: a review," Knowledge and Information Systems, vol. 57, pp. 245-285, 2018.
[3] M. Weber, G. Domeniconi, J. Chen, D. K. I. Weidele, C. Bellei, T. Robinson, and C. E. Leiserson, "Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics," ArXiv Preprint ArXiv:1908.02591, 2019.
[4] T. N. Kipf, M. Welling, "Semi-supervised classification with graph convolutional networks," ArXiv Preprint ArXiv:1609.02907, 2016.
[5] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," ArXiv Preprint ArXiv:1710.10903, 2017.
[6] W. Hamilton, Z. Ying, J. Leskovec, "Inductive representation learning on large graphs," Advances in Neural Information Processing Systems, vol. 30, pp. 1025-1035, 2017.
[7] B. Jiang, Z. Zhang, D. Lin, J. Tang, and B. Luo, "Semi-supervised learning with graph learning-convolutional networks," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CA, USA, 2019, pp. 11313-11320.
[8] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, and Z. Zhang, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," ArXiv Preprint ArXiv:1909.01315, 2019.
[9] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, and C. Leiserson, "Evolvegcn: Evolving graph convolutional networks for dynamic graphs," in Proceedings of the AAAI Conference on Artificial Intelligence, 2020, vol. 34, no. 4, pp. 5363–5370.
[10] S. Hochreiter, J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.