A Dynamic Graph Convolutional Network for Anti-money Laundering

Tianpeng Wei[1], Biyang Zeng[1], Wenqi Guo[1], Zhenyu Guo[2], Shikui Tu[1(✉)], and Lei Xu[1(✉)]

[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
{tushikui,leixu}@sjtu.edu.cn

[2] Big Data & Artificial Intelligence Laboratory, Industrial and Commercial Bank of China, Shanghai 200120, China

Abstract. Anti-money laundering (AML) is essential for safeguarding financial systems. One critical way is to monitor the tremendous daily transaction records to filter out suspicious transactions or accounts, which is time consuming and requires rich experience and expert knowledge to construct filtering rules. Deep learning methods have been used to model the transaction data by graph neural networks, and achieved promising performance in AML. However, the existing methods lack efficient modeling of the transaction time stamps, which provide important discriminative features to efficiently recognize the accounts participating money laundering. In this paper, we propose a dynamic graph attention (DynGAT) network for detecting suspicious accounts or users, which are involved in illicit transactions. The daily transaction records are naturally constructed as graphs, by considering the accounts as nodes and the transaction relationship as edges. To take the time stamps into account, we construct one transaction graph from the records within every time interval, and obtain a temporal sequence of transaction graphs. For every graph in the sequence, we not only compute the node embeddings by a vanilla graph attention network, but also explicitly develop a time embedding via a position-encoding block. Our method further captures the dynamics of the graph sequence through a multi-head self attention block on the sequence of concatenations of node embeddings and time embeddings. Moreover, we train the model by a weighted cross entropy loss function to tackle the sample imbalance problem. Experiments demonstrate that our method outperforms the existing ones in AML task.

Keywords: Anti-money laundering · Graph attention network · Dynamic graph

1 Introduction

Money laundering aims to legalize illegal profits, and it is a severe financial crime. It may cause direct financial losses, or provide concealed funds to criminals for illegal activities. For example, according to [8], the amount of money laundering between Nigeria and USA is up to one trillion US dollars per year through financial crimes, and the money

D.-S. Huang et al. (Eds.): ICIC 2023, LNAI 14090, pp. 493–502, 2023.
https://doi.org/10.1007/978-981-99-4761-4_42

further diverts to illegitimate applications like drug trafficking. However, the detection of money laundering crimes is complex because the illegal clients and transactions are in extreme considering the number of total clients and relevant transactions. Also, illegal users try to cover up himself with normal transactions and changes money laundering techniques rapidly. So, it is hard to conclude a recognition pattern for long term. Besides, there is few public data set released by companies or banks as the real transaction records are confidential, which further limits the research on anti-money laundering problem. Considering the harmful consequences of money laundering, many efforts are put to research on how to identify money laundering behaviors and locate relevant suspects in order to prevent them.

The identification methods evolve with the advance of computer science technology, from manual hard-code rules to automatic detection with machine learning strategies. Traditional machine learning methods, e.g., logistic regression, support vector machine, multilayer perceptron (MLP), have been used in AML to reduce the human burden [3, 5]. One limitation of these works is that they still heavily rely on the manual computation of certain features, which demands expert knowledge. For example, to obtain the characteristics of the trading counterparties, one needs to compute the number of different individuals one has transactions with. Such important features do not exist in original data, and it is difficult to calculate an appropriate one especially when considering high-order counterparties.

With the recent success of deep learning, deep learning methods have been developed for the AML tasks [7]. For example, a deep autoencoder was developed in [10] to identify anomalies in import and export data of Brazil and determine the fraud possibilities of exporters. Recently, graph neural network models (GNNs) [4] became popular for the AML tasks, because the financial transactions are naturally network data. Specifically, the construction from transaction records to graph is quite simple. The entity (i.e., the account or its holder) is taken as a node, and a flow of currency between two entities defines an edge between the corresponding nodes. All the information of the transaction or the entities can be constructed as initial features for edges or nodes. The overall idea of GNN is to aggregate the information of neighbor vertices in graph, and update the representation of the current vertex. Thus, it is possible to have an end-to-end pipeline for AML, and the money laundering patterns may be automatically detected and learned to improve the efficiency and performance of AML. In an attempt made by [12], the authors suggested that, before running traditional machine learning methods like random forest, it is potentially beneficial to augment the node features with the embeddings computed from graph convolutional networks (GCNs).

It is noted that financial transactions naturally come with time stamps. Actually, the time stamps are very important when taking into account to construct the transaction graphs. See Fig. 1 for example. The records of scenario (a) may be normal daily transactions, while the records of scenario (b) are suspicious with money laundering patterns. If ignoring the time stamps, the transaction graphs for the two scenarios are the same in topology. If using one day as the time interval, the constructed graph sequences become discriminative in representing money laundering patterns. The work in [12] demonstrated that temporal GCN models, e.g., EvolveGCN [9], consistently outperform the GCN on static graphs, because the temporal evolution of transactions provides additional

information for the evaluation of the risk of the money laundering, which is lost in static graphs. Specifically, EvolveGCN [9] cut transaction graphs to different snapshots by time, and trained a recurrent neural network (RNN) to generate the weight matrix for the GCN in every time step. There are also other dynamic graph models. DyGCN [2] proposed a new aggregation method for GCN to update the node features by time. By defining information changes as the variation of neighbors embedding, the node feature in next time step is aggregated as a weighted sum of that in current time step and its information changes. Lian Yu [15] uses a three-layer graph attention network (GAT) to generate node embeddings for each subgraph, and random forest is used for classification with the input of previous embeddings. Temporal-GCN [1] seeks to learn temporal information with an additional long-short term memory (LSTM) layer. The output of LSTM layer is then passed to two layers of TAGCN, which is a variant of GCN, to obtain temporal node embeddings. A function is defined in [13] to assign weights to each graph in the sequence, with the most recent graph has more influence on the final decision. TGAT [14] adopted the idea of position encoding in natural language processing, and proposed an encoding function to generate time feature for networks to learn sequential information.

TRANSACTION RECORDS OF SCENARIO (a)

DATE	TRANSFEROR	PAYEE	AMOUNT ($)
July 1st, 4:00 PM	A	B	2000
July 1st, 5:00 PM	B	A	1000
July 20th, 5:00 PM	B	A	500
July 25th, 5:00 PM	B	A	500

TRANSACTION RECORDS OF SCENARIO (b)

DATE	TRANSFEROR	PAYEE	AMOUNT ($)
July 1st, 3:33PM	B	A	500
July 1st, 3:40PM	B	A	1000
July 1st, 3:46PM	A	B	2000
July 1st, 3:55PM	B	A	500

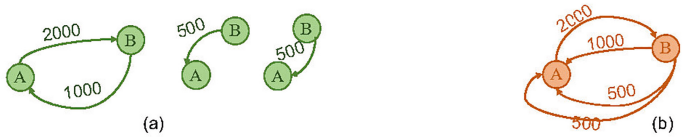

Fig. 1. The transaction time stamps may result in different constructed graphs for a certain time interval. Here, one day has been used as the time interval in both (a) and (b). Notice that if ignoring the time stamps or using one month as the time interval, the topology of the two constructed graphs are the same.

In this paper, we propose a novel dynamic graph attention (DynGAT) network for the AML task. We transform the transaction data into a graph sequence which consist of snapshots from different time clip. Then, the temporal patterns are explicitly exposed in the graph sequence, and can be modeled by a multi-head attention block on the node embeddings of the transaction graphs at each time interval. The node embeddings are computed by a graph attention network [11] which introduces attention mechanism in aggregation process, and learns to assign different weights to the neighbors so as to focus on certain influential entities. Moreover, the dynamic order of the graph sequence is computed as a time embedding via an encoding block, and is concatenated into the node embeddings. It strengthens the discriminative learning of deep representations between the normal transactions and the illicit transactions.

The contribution of the paper can be summarized as two points:

1. The paper proposes a new method to the extraction and application of temporal sequential information from dynamic transaction graphs with the purpose of identification of illegal clients. Particularly, the experiment shows the proposed method improves the performance in AML tasks compared with the static and reference dynamic methods.
2. Instead of traditional RNN-based method, the paper models temporal information explicitly through time encoder, and generates node embeddings for each subgraph. The analysis of which time period contributes the most to the recognition of illegal clients becomes possible by studying the heatmap of attention mechanism. Therefore, the proposed model is able to provide certain explainability for the regulation review of financial institutions.

2 Method

2.1 An Overview of DynGAT

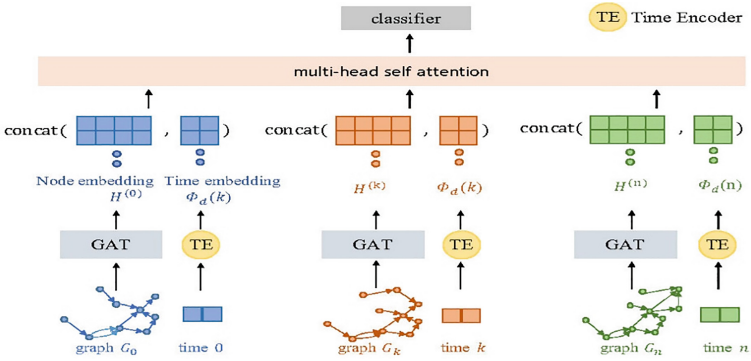

Fig. 2. An overview of the proposed DynGAT. For graphs at different time points, we employ a time encoder (TE), a continuous function $\Phi_d(k)$ that maps time to a vector space, to extract explicitly the temporal order information of the transaction. This time embedding is then concatenated with node embeddings $\boldsymbol{H}^{(k)} = \{h_1{}^{(k)}, \ldots, h_i{}^{(k)}, \ldots\}$ obtained from a GAT encoder, where $h_i{}^{(k)}\$$ is the GAT embedding of the node v_i for the graph G_k in the sequence. Finally, we use multi-head attention to capture temporal information within the graph sequences.

An overview of the proposed DynGAT is given in Fig. 2. Our method mainly contains five parts, i.e., construction of graph sequence from the raw transaction records, node embeddings via GAT, time embeddings via a time encoder, multi-head self attention to capture the temporal patterns evolving in the sequence of concatenated embeddings of the node and the time, and finally a softmax classifier to predict whether a is suspicious or not. The key contribution of the model lies in the explicit modeling of the temporal patterns on the transaction graph sequence. The learned representations between the normal nodes and the suspicious nodes are more discriminative than the existing methods, and thus the prediction performance on the AML task is improved.

2.2 Constructing the Graph Sequence from the Transaction Records

Given transaction records and client information records, we first define a static transaction graph $G = (\boldsymbol{V}, \boldsymbol{E})$ where a vertex $v_i \in \boldsymbol{V}$ is constructed from a client (either the transferor or the payee of the transactions) and a directed edge $e_k = (v_i, v_j) \in \boldsymbol{E}$ represents a flow of funds from the client v_i to the client v_j. Each transaction has a timestamp, and we use t_k to denote the timestamp of the transaction edge e_k. We record the biggest timestamp value as T_{max} and the smallest one as T_{min}.

To convert static graph into dynamic graph sequence, we use the following formula to determine start time $t_{start}^{(i)}$ and end time $t_{end}^{(i)}$ for the i-th graph in dynamic sequence:

$$t_{start}^{(i)} = \frac{i \cdot (T_{max} - T_{min} + 1)}{n}, t_{end}^{(i)} = t_{start}^{(i)} - 1, \tag{1}$$

where n refers to the number of time clips. Then we define our dynamic graph sequence as $\mathcal{G} = \{G_0, G_1, \ldots G_{n-1}, G_n\}$. Each graph G_i in the sequence is expressed as the follows:

$$G_i = (\boldsymbol{V}, \boldsymbol{E}_i), \boldsymbol{E}_i = \{e_k | t_{start}^{(i)} \leq t_k \leq t_{end}^{(i)}\}, \tag{2}$$

Generally, each graph G_i has a different subset of the transaction edges which happens in the specified time period, but it uses the same set of all nodes and attributions. Usually, the bank accounts are relatively stable, as the events of registering new accounts or closing the old ones are rare. It is reasonable to assume that the accounts and their attributes are unchanged at a certain period.

2.3 Computing the Node Embedding and the Time Embedding

We employ a GAT as a feature extractor from the graph sequence. A node embedding is computed for each graph in the sequence by the GAT. Like most NN variants, GAT is implemented by a message passing process and an aggregation function. The network details of GAT are referred to the original paper [10]. We adapt the standard GAT in this paper to the AML task by integrating he features of the transaction edges. Specifically, for an account v_i after the graph construction, we will have a set of accounts as its neighbors, denoted as $\boldsymbol{N}_i$, in a certain graph G_k. To calculate the node embedding of v_i, GAT first calculates the similarity s_{ij} between the current account v_i and its transaction counterparties v_j in the graph. Then, the similarity is normalized to obtain the attention coefficient α_{ij} for each neighbor.

In this paper for the AML task, the edges of a transaction graph are often attached with additional features, e.g., the amount of the transaction. Hence, the original equation in GAT is generalized to integrate the edge features $\boldsymbol{f}_{ij}$ [6] as follows:

$$s_{ij} = softmax(LeakyReLU(\boldsymbol{a}^T[\boldsymbol{W}\boldsymbol{h}_i || \boldsymbol{W}\boldsymbol{f}_{ij} || \boldsymbol{W}\boldsymbol{h}_j])) \tag{3}$$

where $\boldsymbol{a}$ and $\boldsymbol{W}$ are learnable parameters, the operator "||" denotes the concatenation of the two vectors, and $\boldsymbol{h}_i$ is the embedding of the node v_i. The initialization of $\boldsymbol{h}_i$ is the raw feature vectors of the nodes (e.g., the age of the account).

The node embedding $\boldsymbol{h}_i$ is updated as the weighted sum of the neighbors' representations $\boldsymbol{h}_j$, where the weights are given by the attention coefficient α_{ij} which is computed from s_{ij} in Eq. (3).

The above computed node embeddings $\boldsymbol{h}_i$ is able to encode the topological patterns of the money launderers, but it fails to incorporate the information of the time stamps. To remedy this issue, one may use the time stamps directly as features, but this manner could harm the generalization performance of the model which may overfit the time stamps.

Here, we develop a time encoder module to convert the one-dimension time sequence into time embeddings instead of directly using the time itself. This is a very similar trick as the position encoding in the field of natural language processing. Our time encoding function $\Phi_d(t)$ is adopted from [14]. The input of $\Phi_d(t)$ is a time sequence $\{0, 1, \ldots, n\}$. The time encoder maps each time to a d dimensional vector, and outputs $\{\Phi_d(0), \Phi_d(1), \ldots, \Phi_d(n)\}$.

Finally, we fuse the graph structure information and the transaction time information together by concatenating the node embeddings with the time embedding, i.e.,

$$\overline{\boldsymbol{h}_i}^{(k)} = [\boldsymbol{h}_i{}^{(k)} || \Phi_d(n)] \tag{4}$$

where $\overline{\boldsymbol{h}_i}^{(k)}$ is the final output of the deep representations of the node v_i for the graph G_k in the sequence. It contains not only the topological patterns of the laundering behaviors, but also the timing features of the transactions which are important to detect the money launderer.

2.4 Modeling the Temporal Information by the Multi-head Self Attention Block

The final step is to integrate i's embedding from different timestamps. The embedding set of a node i is marked as $\left\{\overline{h_i}^{(k)} \middle| k \in (0, 1, \ldots, n)\right\}$. A common method in dynamic sequence is to train a classifier to give a risk score for every hidden representation, and apply an average or max pooling to integrate the scores. The drawback of the traditional method is that the sequential information across time steps is lost. For example, an account appears to be normal if it receives money from multiple persons, or transfers money to another account. But if it receives multiple transactions and transfers money a to certain one, it becomes very suspicious.

Self-attention is invented to identify the relevant tokens within a sentence, and shows great ability in modeling the positional information between token with position encoding. Under the time sequential scenario, we can also apply self-attention operation to learn the sequential information of temporal hidden representations across time steps. The inputs of self-attention are three matrices $\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$. To calculate our own $\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$, we reorganize the embedding set to be an embedding matrix $\boldsymbol{Z}$ where the k_{th} column of $\boldsymbol{Z}$ is $(\overline{\boldsymbol{h}_i}^{(k)})^T$. Then, multiply $\boldsymbol{Z}$ with project matrix $\boldsymbol{W}_q$ to acquire the input Query matrix which can be expressed as $\boldsymbol{Q} = \boldsymbol{W}_q\boldsymbol{Z}$. We then have Key matrix $\boldsymbol{K}$ and Value matrix $\boldsymbol{V}$ following the same manner. Let d be the length of $\overline{\boldsymbol{h}_i}^{(k)}$. Then, the integrated

representation matrix $\boldsymbol{F}_i$ of node i can be calculated with the follow equation.

$$\boldsymbol{F}_i = Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^{\boldsymbol{T}}}{\sqrt{d}}\boldsymbol{V}) \tag{5}$$

To stabilize the output, multi-head scheme is usually applied. Generally, multi-head operation is to calculate self attention function for multiple times individually and concatenate different outputs. Then, a projection matrix is used to restore the shape of output matrix.

2.5 Training the Model

The loss function is binary cross entropy function. In AML task, we care more about the minor illegal class. To overcome the imbalance class problem, we assign additional weights during the calculation of loss function. With a higher weight attached to the illegal class, we force the model to focus on the illegal samples.

3 Experiment

3.1 Experiment Setting

The AML task can be treated as a binary classification problem. Each node should be classified as either legal or illegal. In our experiment, a 3-layer MLP will be used as classifier. The dynamic graph sequence will be organized according to the definition of Sect. 2.1, and we set the number of clips to be 5. We set the number of GAT layers to be 2. To solve the class imbalance problem, we assign [0.7, 0.3] as weight parameter of our loss function. Our models are trained on Tesla V100 and the cpu is Intel(R) Xeon(R) Gold 6130 CPU @ 2.10 GHz.

3.2 Dataset

- Bitcoin-Elliptic

  Bitcoin-Elliptic dataset records bitcoin transaction events in 49 time steps. Each node in dataset represents a transaction and each edge is a flow of payment between transactions. 2% of all nodes are labeled as illegal, 21% are labeled as legal and the rest are unclassified. In real banking system, the clients that have not been identified by experts should be treated as normal ones. We follow the routine and view unclassified nodes as legal in our experiment resulting in 4545 illegal nodes and 226224 legal nodes.
- AMLSim-10K

  AMLSim is a synthetic data generator developed by IBM company. It can simulate real banking environment and provides synthetic transaction records and clients' KYC (Know Your Customer) information according to revealed money laundering patterns. We use the default setting to generate this dataset with 12043 clients in total of which 737 are labeled as suspects, and name it as AMLSim-10K (Table 1).

Table 1. Overview of data sets

Dataset	#Nodes	#Node features	#Edges	#Edges features
Bitcoin-Elliptic	203769	166	234355	0
AMLSim-10K	12043	5	197905	2

3.3 Baseline Method

We first choose static GAT to test the performance without any temporal information. Two implicit temporal model evolveGCN and GRU-GCN are also selected as baseline. EvolveGCN uses GCN to generate node embeddings on each subgraph, but does not update the weight matrix of GCN during loss backward stage. Instead, a RNN is trained to update the weight matrix in the next time step with the current weight matrix as input. GCN-GRU is a common method for dynamic graph sequence. It uses a GRU cell to connect the outputs of each time step and uses the output of the final step as the final output. To analyze the effect of time encoder, we create DynGAT-T by removing time encoder module from DynGAT.

3.4 Performance

The result from Table 2 shows our method outstands the baseline methods which proves the effectiveness of our method. In Elliptic data set, the performance of our method in all four metrics is best among all methods. In AMLSim-10K data set, our method achieves the best precision and ROC-AUC. The static GAT has an extremely high recall, because it tends to classify everyone as illegal accounts, which means it does not successfully learn a pattern to divide the two classes. Our method, however, manages to maintain a good balance on them.

Table 2. Experiment result on Elliptic and AMLSim-10K data sets

	Elliptic				AMLSim-10K			
Method	precision	recall	F1	ROC-AUC	precision	recall	F1	ROC-AUC
Static GAT	0.603	0.494	0.543	0.952	0.056	**0.985**	0.105	0.379
GCN-GRN	0.229	0.413	0.294	0.777	0.137	0.311	0.190	0.560
evolveGCN	0.235	0.161	0.191	0.769	0.097	0.322	0.149	0.518
DynGAT-T	0.516	0.547	0.531	0.923	0.284	0.470	**0.354**	0.653
DynGAT	**0.605**	**0.654**	**0.628**	**0.962**	**0.421**	0.242	0.308	**0.695**

The PR curves in Fig. 3 show that generally GAT methods perform better than GCN based methods in Elliptic data set. Static GAT even performs better than other temporal baseline methods. We conclude that the graph constructed from Elliptic is spare, the

average degree of each node is 1.1. When cutting into different graphs, each subgraph becomes more sparser, and the other GCN-based methods fails to extract useful patterns in each subgraph. Because DynGAT-T cannot use time encoding to learn sequential pattern and it sees less global information than static GAT, static GAT is even able to perform better than DynGAT-T.

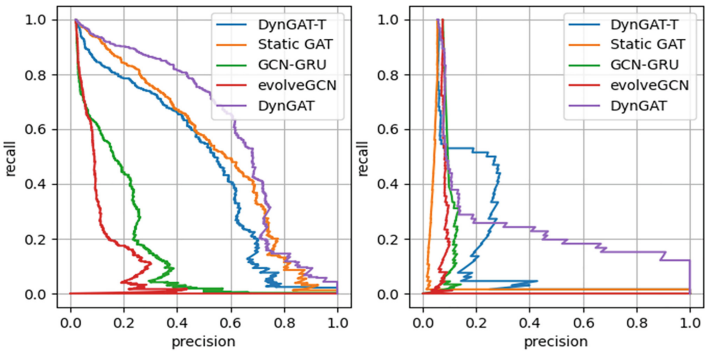

Fig. 3. PR Curve of Elliptic data set (left) and AMLSim-10K (right)

AMLSim-10K is a more connected graph than Elliptic. All methods suffer a loss of performance in AMLSim-10 while the DynGAT and DynGAT-T stand out. Static GAT is not able to learn patterns from such a complicated graph, as information from different time period is mixed up. For example, if an account receives from 1 account and 11 accounts in two days continuously, it seems suspicious for abnormal increase of transaction times. On average, it has 5 transactions per day which seems to be normal. In this case, the temporal models perform better.

4 Conclusion

The paper presents a new dynamic graph learning model called DynGAT for the anti-money laundering task. We expose the temporal information via constructing a sequence of dynamic graphs from the transaction records. With the employment of a time encoder to calculate the temporal hidden representation for nodes in dynamic graphs, and fuse with the node embeddings computed by GAT for the structural behavior of money laundering activities. We devise a multi-head attention to further capture the sequential information of the temporal hidden representations from different time steps. The learned representations of the users by our method are more discriminative between the normal users and the money launderers. The proposed method explicitly encodes the time variable as part of the node feature and applies attention mechanism in both graph encoder and the fusion process, which helps to explore the importance and contribution of time in AML identification by analyzing heat maps of attention mechanism. Experiments show the effectiveness of our method and outperforms the existing methods in detecting the suspicious users or accounts.

Acknowledgement. This work was supported by Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), and ICBC (grant no. 0010100005202201O6). Shikui Tu and Lei Xu are co-corresponding authors.

References

1. Alarab, I., Prakoonwit, S.: Graph-based LSTM for anti-money laundering: experimenting temporal graph convolutional network with bitcoin data. Neural Process. Lett. **55**(1), 689–707 (2023)
2. Cui, Z., Li, Z., et al.: Dygcn: Efficient dynamic graph embedding with graph convolutional network. IEEE Trans. Neural Netw. Learning Syst. (2022)
3. Feng, Y., Li, C., et al.: Anti-money laundering (AML) research: a system for identification and multi-classification. In: Ni, W., Wang, X., Song, W., Li, Y. (eds.) Web Information Systems and Applications, pp. 169–175. Springer International Publishing, Cham (2019)
4. Gori, M., Monfardini, G., Scarselli, F.: A new model for learning in graph domains. In: Proceedings of the 2005 IEEE International Joint Conference on Neural Networks, vol. 2, pp. 729–734. IEEE (2005)
5. Jullum, M., Løland, A., et al.: Detecting money laundering transactions with machine learning. J. Money Laundering Control **23**(1), 173–186 (2020)
6. Kamiński, K., Ludwiczak, J., et al.: Rossmann-toolbox: a deep learning-based protocol for the prediction and design of cofactor specificity in rossmann fold proteins. Briefings in Bioinform. **23**(1), bbab371 (2022)
7. Kute, D.V., et al.: Deep learning and explainable artificial intelligence techniques applied for detecting money launderinga critical review. IEEE Access **9**, 82300–82317 (2021)
8. Olujobi, O.J., Yebisi, E.T.: Combating the crimes of money laundering and terrorism financing in Nigeria: a legal approach for combating the menace. J. Money Laundering Control **26**(2), 268–289 (2023)
9. Pareja, A., Domeniconi, G.,et al.: EvolveGCN: evolving graph convolutional networks for dynamic graphs. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 5363–5370 (2020)
10. Paula, E.L., Ladeira, M., et al.: Deep learning anomaly detection as support fraud investigation in Brazilian exports and anti-money laundering. In: Proceedings of the 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 954–960. IEEE (2016)
11. Veličković, P., et al.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
12. Weber, M., Domeniconi, G., Chen, J., Weidele, D.K.I., Bellei, C., Robinson, T., Leiserson, C.E.: Anti-money laundering in bitcoin: experimenting with graph convolutional networks for financial forensics. arXiv preprint arXiv:1908.02591 (2019)
13. Wu, B., Liang, X., et al.: Improving dynamic graph convolutional network with fine-grained attention mechanism. In: ICASSP 2022–2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3938–3942. IEEE (2022)
14. Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., Achan, K.: Inductive representation learning on temporal graphs. arXiv preprint arXiv:2002.07962 (2020)
15. Yu, L., Zhang, N., Wen, W.: Abnormal transaction detection based on graph networks. In: Proceedings of the 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC), pp. 312–317 (2021)