Eigenvector-based Graph Neural Network Embeddings and Trust Rating Prediction in Bitcoin Networks

Pin Ni*
Institute of Finance and Technology, University College London
London, UK
pin.ni.21@ucl.ac.uk

QiAo Yuan*
Department of Computer Science, University of Liverpool
Liverpool, UK
q.yuan11@liverpool.ac.uk

Raad Khraishi
Institute of Finance and Technology, University College London
London, UK
raad.khraishi@ucl.ac.uk

Ramin Okhrati
Institute of Finance and Technology, University College London
London, UK
r.okhrati@ucl.ac.uk

Aldo Lipani
Institute of Finance and Technology, University College London
London, UK
aldo.lipani@ucl.ac.uk

Francesca Medda
Institute of Finance and Technology, University College London
London, UK
f.medda@ucl.ac.uk

ABSTRACT

Given their strong performance on a variety of graph learning tasks, Graph Neural Networks (GNNs) are increasingly used to model financial networks. Traditional GNNs, however, are not able to capture higher-order topological information, and their performance is known to degrade with the presence of negative edges that may arise in many common financial applications. Considering the rich semantic inference of negative edges, excluding them as an obvious solution is not elegant. Alternatively, another basic approach is to apply positive normalization, however, this also may lead to information loss. Our work proposes a simple yet effective solution to overcome these two challenges by employing the eigenvectors with top-k largest eigenvalues of the raw adjacency matrix for pre-embeddings. These pre-embeddings contain high-order topological knowledge together with the information on negative edges, which are then fed into a GNN with a positively normalized adjacency matrix to compensate for its shortcomings. Through comprehensive experiments and analysis, we empirically demonstrate the superiority of our proposed solution in a Bitcoin user reputation score prediction task.

CCS CONCEPTS

• **Applied computing → Secure online transactions**.

KEYWORDS

Graph Neural Network, Eigenvector, Digital Transactions, Blockchain, Rating Prediction

*Both authors contributed equally to this research.

ICAIF '22, November 2–4, 2022, New York, NY, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9376-8/22/11…$15.00
https://doi.org/10.1145/3533271.3561793

ACM Reference Format:
Pin Ni, QiAo Yuan, Raad Khraishi, Ramin Okhrati, Aldo Lipani, and Francesca Medda. 2022. Eigenvector-based Graph Neural Network Embeddings and Trust Rating Prediction in Bitcoin Networks. In *3rd ACM International Conference on AI in Finance (ICAIF '22), November 2–4, 2022, New York, NY, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3533271.3561793

1 INTRODUCTION

Despite a massive increase in its popularity over recent years, Bitcoin, a digital currency based on blockchain technology, has been continuously criticized for its role in facilitating financial crimes [5, 43]. Its innovative characteristics (such as decentralization and anonymity) that have helped promote its image as a disruptor of modern financial systems, have also attracted various criminal activities including money laundering, blackmail, trafficking, and scams [50]. Recent literature has begun to address these issues by investigating the risks associated with bitcoin transaction networks and exploring techniques to detect malicious activities [1, 54].

Deep learning, for instance, is one such technique that has seen recent success across a number of problem domains and is increasingly being researched in academia and industry to augment or replace traditional rules-based monitoring systems (see [15] for an overview). More generally, deep learning is widely used for anomaly detection [36], fault or disease diagnosis [26, 29, 34, 51], as well as pattern, sequence recognition [9, 19, 32] and natural language processing [30, 31, 33, 35, 39–41]. Deep learning research within the Bitcoin literature, however, has focused on forecasting future prices and returns [18] with relatively few papers exploring how it can be used to identify financial crimes [3, 6].

Recently, common deep learning architectures have been extended to leverage the topological properties of data. For example, Graph Convolutional Networks (GCN) leverage the concept of weight sharing and can capture complex non-linear relationships between nodes. As a feature extractor for graphs, these networks may be adapted to any tasks involving graph-structural data such as protein identification [45], collaborative filtering [46] and text summarization [52]. These architectures are potentially appropriate for financial transaction data which can be modeled and analyzed using graphs with interconnected users. For example, node identification can be used to identify whether a transaction is legal, and link prediction can determine whether two financial entities have a specific relationship. These techniques may also be used for decentralized financial networks such as Bitcoin, in which transaction information exists on each node in the form of distributed storage and the nodes are linked to each other, which can be regarded as an explicitly weighted signed directed network. Several recent studies have started to explore how these models may be used to detect illicit activities within the Bitcoin network [27, 44, 53].

Our approach builds on these recent papers by exploring how classical neural learning and Graph Neural Networks (GNNs) may be used for malicious or suspicious behavior detection in the Bitcoin transaction network or other similar financial networks requiring supervision. We formulate a prediction task using mutual rating scores among member nodes provided by the Bitcoin OTC (Over-The-Counter) and Bitcoin Alpha platforms. These scores range from -10 (total distrust) to +10 (total trust) and the magnitude represents the degree of distrust or trust, respectively. Due to the anonymity of Bitcoin transactions, our model relies solely on these mutual ratings of transactions to evaluate the reputation of nodes. We demonstrate that with this network, for each node, we are able to generate high-quality embeddings that may be used to accurately identify reputation ratings as well as to develop further practical financial regulatory applications including fraud detection. We also extend the literature by exploring several modeling approaches and practical challenges, including normalization, eigenvector extraction, and unsupervised pre-training of the embeddings.

2 RELATED WORK

2.1 Graph Neural Networks

Graph Neural Networks (GNNs) are a family of neural network architectures that may be applied to non-Euclidean data structures such as graphs [55]. These networks can be categorized as either spectral or spatial-based.

Spectral-based GNNs map the graph data to the spectral domain with Fourier transform, perform convolution in the spectral domain, and then return the data back to the spatial domain with an inverse Fourier transform. In an early paper, [11] treat the convolution kernel as a n-dimensional trainable parameter since the convolution kernel is concerned with all the eigenvalues of the Laplacian matrix. This early approach was computationally expensive and did not consider local topological information. [10] extend this approach and approximated the convolution kernel with a k-order truncated polynomial, reducing the computation complexity and importing k-order local structural knowledge. [20] further simplified the convolution kernel with first-order truncated polynomials, where each node merely observes its first-hop neighbors at each layer. With this approach, the receptive field of each node can still be expanded by stacking GNN layers. Although still classified as spectral-based, this first-order approximation GCN is more closely related to spatial-based GNNs with first-order neighborhood aggregation.

Spatial-based GNNs introduce convolution through local neighborhood aggregation where each node updates its representation by iteratively aggregating its neighbor's information. For example, GraphSAGE [14] introduce several aggregation operators such as "mean", "max pool", "sum", and "Long Short-Term Memory (LSTM)" to allow the GNN to be applied to various tasks. To allow for dynamic edge weights, [45] use an attention mechanism to dynamically adjust the edge weights, encouraging each node to focus on its critical neighbors.

In our experiment, outlined in Section 4.1, we assume access to the edge weights and evaluate the performance of three different approaches on a downstream task: a simplified version of GCN without an activation function (referred to as Simplifying graph Convolutional Networks), a first-order approximation GCN (which we refer to as GCN), and GraphSAGE.

2.2 Applications of GNNs in Finance

There is rich literature on the application of deep learning and graph learning algorithms in finance [8, 24, 25, 38]. For example, [47] introduce scalable graph convolutional neural networks for forensic analysis of financial data to support anti-money laundering (AML). They used a large synthetic graph they created to conduct preliminary experiments and highlighted the promise of graph deep learning in the field of financial crime detection. In a more recent paper, [48] introduce a time series graph dataset that contains many Bitcoin transactions (nodes) and directed payment flows (edges) which they use in an experiment to identify illicit transactions. They compared their results using different variants of the graph convolutional network (GCN) against more traditional approaches such as random forests, multilayer perceptrons, and logistic regression models. The authors found that the GCN models are able to improve performance and leverage the topological structure of the data. They further provide a model that allows for navigating the graph and observing the performance of anti-money laundering activities over time.

2.3 Network Embedding Approaches

Node embeddings are often estimated using either GNNs, matrix factorization (MF), or random walks (RW). GNN-based approaches iteratively update node embeddings through neighborhood aggregation and as such most GNNs can be used for node embeddings. MF-based methods factorize an adjacency matrix (which may be a matrix other than the Laplacian) as two identical low-rank latent matrices, which can be treated as the embedding matrix. For example, [17] estimates node embeddings by minimizing the Euclidean distance between the multiplication of two latent matrices and the node similarity matrix. [16] extend this and import local structure information into the similarity matrix, such that each node is only connected with its k-nearest neighbors. The work by [4] extends this further by utilizing auxiliary information (i.e., node labels) to improve the training such that the node embedding is encouraged to maintain local topological information and preserve class separability.

RW-based approaches on the other hand draw inspiration from skip-gram language models, which perform random walks upon a graph and sample a list of paths. In the language analogy, each path is treated as a sentence and each node is regarded as a word. The node embeddings are then obtained by maximizing the concurrency probability of nodes within a window in a path. Differences in approaches to how paths and candidate nodes are chosen have given rise to several different RW-based algorithms. For example, the DeepWalk algorithm introduced by [37] employs an unbiased random walk to select candidate nodes. LINE [42], on the other hand, treats the first and second-hop neighbors as a node's relevant context, which is analogous to the Breadth-First Search (BFS), and node2vec [13] offers a balance between BFS and Depth-First Search (DFS).

While all the methods described may be used to estimate graph embeddings, there are some trade-offs between them. For example, MF-based methods fail to exploit the structural information of graphs. While GNNs and RW-based methods are able to leverage this information, they are weak in addressing negative edges. For GNNs, current research tends to abandon the negative edges to retain the graph homophily [46], regardless of the rich semantic meaning of negative signals.

We aim to improve upon these issues in our current work through embedding methods that can both accommodate negative edges and leverage the global topological information. First, we design a loss function based on the objective of MF-based methods. We also reconstruct the weighted adjacency matrix by multiplying the node embedding matrix and utilized the reconstruction loss to guide the network. Then, to inject local topological information into node embedding, we employ GNNs as the intermediary component. Finally, to incorporate the semantics of negative edges and high-order structural information, we adopted top-k largest eigenvectors as the node pre-embedding. For this step, we explore both DeepWalk and node2vec as pre-embedding methods.

3 PRELIMINARIES

3.1 Graph Neural Networks

A graph is a structure that may be used to represent interconnected data points (e.g., a network of users connected by financial transactions). A graph G can be defined as $G = (V, E, \mathbf{A})$, where V is the node-set, E is the edge set and, $\mathbf{A} \in R^{n \times n}$ is an adjacency matrix (with n equal to the number of nodes) describing the connections between nodes. Normally, $\mathbf{A}_{ij} = 1$ if there is a link from node i to node j, otherwise, $\mathbf{A}_{ij} = 0$. In some cases, $\mathbf{A}$ may be a weighted matrix, where $\mathbf{A}_{ij} \neq 0$ not only implies that there is a link from node j to node i but also indicates the properties of this link. Each node $v_i \in V$ may also contain additional information such as a vector x_i of features describing its properties.

Graph Neural Networks (GNNs) is a state-of-the-art graph learning technique that has grown in popularity over recent years. They incorporate the topological information into node representations with a message-passing mechanism where each node receives information from its neighbors to update its own representation and sends its information to neighbors at the same time [12]. As shown in Figure 1, the input of a GNN is a graph with node features and the output is a graph with an updated node representation. In each hidden layer, each node aggregates its neighbors' information once and then updates its own representation by performing a non-linear transformation upon the aggregated information. Formally, the forward propagation of each GNN layer can be written as:

$$\mathbf{H}^{l+1} = \sigma(Agg(F(\mathbf{A}), \mathbf{H}^{l}, W^{l})) \tag{1}$$

where σ is an activation function, Agg is an aggregation operation, $F(\mathbf{A})$ is an improved adjacency matrix, $\mathbf{H}^l$ is the node representation matrix at the l-th layer, and W^l is a trainable parameter matrix at the l-th layer. The design of the aggregation operation and the improved adjacency matrix is where various GNNs differ from each other.

Note that although GNNs are often categorized as spectral-based and spatial-based, the introduction of a first-order approximation of the convolutional kernel by [20] has made spectral-based GNNs equivalent to the first-order neighborhood aggregation in the spatial domain.

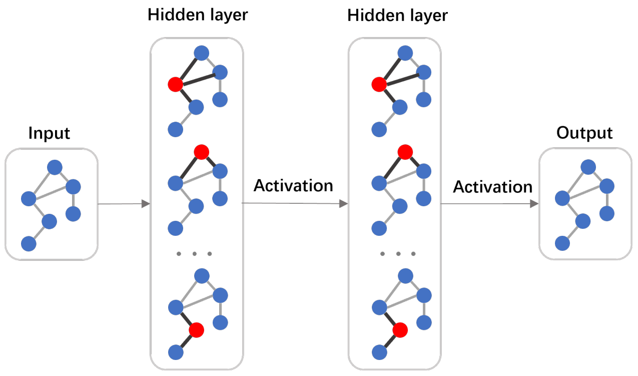

Figure 1: A schematic view of generic graph neural networks.

3.2 Adjacency Matrices

Several approaches have been introduced to improve the adjacency matrix used by a GNN. For example, many GNNs introduce a self-loop to the adjacency matrix to prevent each node from forgetting its own information, acquiring $\tilde{\mathbf{A}} = \mathbf{A} + I$ where I is the identity matrix. Some GNNs [20, 49] employ a normalized adjacency matrix defined as:

$$F(\tilde{\mathbf{A}}) = \mathbf{D}^{-\frac{1}{2}}\tilde{\mathbf{A}}\mathbf{D}^{-\frac{1}{2}} \tag{2}$$

where $\mathbf{D}$ is the degree matrix of $\tilde{\mathbf{A}}$. In this case, the weight of the link from node i to node j is normalized by their degrees as:

$$F(\tilde{\mathbf{A}})_{ij} = \frac{\tilde{\mathbf{A}}_{ij}}{\sqrt{d_i}\sqrt{d_j}} \tag{3}$$

where d_i is the degree of node i. Others have employed a random walk adjacency matrix [2] defined as:

$$F(\tilde{\mathbf{A}}) = \mathbf{D}^{-1}\tilde{\mathbf{A}} \tag{4}$$

where the weight of the link from node i to node j is normalized by the degree of node i.

3.3 DeepWalk & Node2vec

DeepWalk [37] and Node2vec [13] are two node embedding methods drawing inspiration from language models. To estimate node embeddings, they perform random walks among graphs to sample a list of paths. The list of paths is regarded as the corpus of a language model such as word2vec [28], where each path is a sentence, and each node is a word. The node embeddings are then the by-product of maximizing the node concurrence possibility in these paths. DeepWalk and Node2Vec differ in their strategies to generate the paths. Given a path c where c_i denotes the i-th node u in the path, in DeepWalk, the possibility to select v as the $i+1$-th node of c is defined by:

$$p(v = c_{i+1}|u = c_i) = \begin{cases} \frac{\pi_{uv}}{z} & (u, v) \in E \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where π_{uv} is the transition probability from node u to v and z is a normalization constant. This method is BFS preferred, which tends to visit the nodes that are close to the previous node. The node2vec methods imports parameters to adjust the preference to BFS or DFS, whose sampling protocol is defined by:

$$p(v = c_{i+1}|u = c_i) = \begin{cases} \frac{\pi_{uv}}{pz} & d_{uv} = 0 \\ \frac{\pi_{uv}}{z} & d_{uv} = 1 \\ \frac{\pi_{uv}}{qz} & d_{uv} = 2 \end{cases} \tag{6}$$

where d_{uv} denotes the shortest path between node u and node v, $d_{uv} = 0$ denotes that v is visited before u (thus if p is small, the model is likely to revisit the observed node), $d_{uv} = 2$ indicates that u and v are 2-hop neighbors (thus if q is small, the model tends to visit the 2-hop neighbors of the previous node, which is analogous to DFS search).

4 METHODOLOGY

4.1 Problem Definition

We define our graph to be a rating network where nodes are Bitcoin users and the edge weights are the rating scores from one user to another. Negative rating scores indicate "distrust" between users while positive implies "trust" and the magnitude of the score denotes the degree of "distrust" or "trust" . Since Bitcoin users are anonymous, there is no explicit user information that may be used to assess whether a user is trustworthy, and we rely solely on the rating score between users to evaluate a user's reputation. A user with a sizable proportion of negative edges is likely to be malicious, and thereby other users may seek to avoid trading with them. We aim to generate a high-quality embedding of each user node that could be used to precisely model its reputation, thus facilitating a variety of downstream applications such as fraud detection. To evaluate the quality of generated node embeddings, we mask a part of edges and perform edge weight prediction tasks with the obtained embeddings.

4.2 Negative Edges and Degradation of Traditional GNN Performance

Traditional GNNs are designed based on the homophily (or assortativity) assumption that the representation of a node should be similar to its neighbors [56]. Through iterative neighbor aggregation, the representation of each node becomes close to its neighbors. When a node i at the l-th layer updates its representation, its desirable representation at the next layer should minimize its distance to its neighbors, which can be defined as:

$$D(v_i) = \sum_{j \in N_i} ||h_i^{l+1} - h_j^l||_2^2 \tag{7}$$

where N_i is the first-order neighbors of node i and h_i^{l+1} is the representation of node i at $l+1$-th layer. Here for simplicity, we set the edge weight to 1. Incorporating a self-loop, each node is expected to approach its neighbors and keep close to its original representation simultaneously. As such, the distance is transformed to:

$$D(v_i) = ||h_i^{l+1} - h_i^l||_2^2 + \sum_{j \in N_i} ||h_i^{l+1} - h_j^l||_2^2 . \tag{8}$$

Therefore, to obtain the desirable representation h_i^{l+1}, the objective is to minimize $D(v_i)$. It is intractable to regard all the nodes as dynamic objects in the graph. Based on the mean-of-field approximation, we can fix other nodes and merely focus on the update of v_i. Hence, $D(v_i)$ may be further transformed to a multi-variable Gaussian distribution which is a convex function. By setting its derivative as 0, we can get the closed-form solution of h^{l+1}:

$$h_i^{l+1} = \frac{h_i^l + \sum_{j \in N_i} h_j^l}{1 + \sum_{j \in N_i}} = \frac{h_i^l + \sum_{j \in N_i} h_j^l}{1 + d_i} \tag{9}$$

which is in the form of neighborhood aggregation.

So far, we have shown that in a graph with all positive edges, neighborhood aggregation strengthens the graph homophily. Now, we incorporate negative edges and rethink what neighborhood aggregation does in graphs with negative edges (for simplicity, we set positive edge weight as 1 and negative edge weight as -1). Furthermore, we incorporate the intuition that a node should keep away from its negatively connected neighbors. For instance, in the bitcoin rating network, a user is likely to avoid approaching another user that it distrusts and tends to assign high scores to users with a similar reputation. Based on this intuition, to obtain the representation h^{l+1}, the objective is to minimize the distance of the node to its positively connected neighbors while maximizing the distance to its negatively connected neighbors, which amounts to minimizing the following $D(v_i)$:

$$D(v_i) = ||h_i^{l+1} - h_i^l||_2^2 + \sum_{j \in N_{i+}} ||h_i^{l+1} - h_j^l||_2^2 - \sum_{j \in N_{i-}} ||h_i^{l+1} - h_j^l||_2^2 \tag{10}$$

where N_{i+} is the positively connected first-order neighbors of node i and N_{i-} is the negatively connected ones. By setting its derivative as 0, we can also get the update rule:

$$h_i^{l+1} = \frac{h_i^l + \sum_{j \in N_{i+}} h_j^l - \sum_{j \in N_{i-}} h_j^l}{1 + d_i^+ - d_i^-} . \tag{11}$$

where d_i^+ is the positive degree of node i and d_i^- is the negative degree. However, we cannot guarantee the convexity of $D(v_i)$ in this case. The second-order derivative of $D(v_i)$ is $1 + d_i^+ - d_i^-$, which can be positive or negative. If it is positive, then $D(v_i)$ is convex and following the above update rule, it would approach the minimum value. If the second-order derivative is negative, then $D(v_i)$ is concave and following the above update rule, it would approach the maximum value, which contradicts the above intuition. Therefore, the neighborhood aggregation is unstable in the graph with negative edges. The effectiveness of neighbor aggregation depends on the ratio of positive and negative links of a node. Neighborhood aggregation does not help a node if it has more negative linkages than positive edges.

4.3 The Framework

As discussed in the preceding sections, there are two main challenges that our approach needs to address to be able to effectively model Bitcoin rating networks: 1) the lack of available node features; 2) the presence of negative ratings which may reduce the performance of traditional GNNs.

To address the first challenge, we separately generate node embeddings before feeding them into the GNN. A common solution is to represent each node with one-hot encoding, however, this ruins the permutation equivariance of GNNs by encoding sequential node ids and dramatically increases the dimensionality of the problem. Other alternatives such as random initialization or degree encoding are not able fully to exploit the structural information of the graph. Instead, we employ efficient node embedding methods that incorporate topological information before training a GNN.

Note that although GNNs can produce node embeddings that are topologically aware, we still use a separate embedding step for two main reasons. First, traditional GNNs may suffer from an over-smoothing problem, which limits the optimal depth of GNN networks to 2 or 3 layers [7]. Hence, the output representation of each node merely learns the 2 or 3-order local structural knowledge. It is beneficial to integrate global or high-order topological information into the input node embeddings of GNNs. Second, is that because the performance of GNNs is perturbed by negative edges. We may normalize the edge weights to a positive interval such as [0, 1] following the work [23], however, this approach may induce information loss as we treat all the negative links as positive ones. Through neighborhood aggregation, the network would also narrow the discrepancy between one node with another one that it distrusts by not fully exploiting the negative signals. Therefore, it is also beneficial to incorporate the negative signals into the input node embeddings of the GNNs.

As such, we require a node embedding method that can integrate the global (higher-order) topological information and the negative edge weights. Global topological information requires that each node be capable of observing all the other nodes in the graph. Based on the messaging passing mechanism, it amounts to sending the information of a node to all the other nodes and incorporating the information of all the other nodes into each node. The message passing process among a graph with feature matrix X may be absorbed into the multiplication between the random walk adjacency matrix $\mathbf{R} = \mathbf{D}^{-1}\tilde{\mathbf{A}}$ and $X^k = \mathbf{R}^k X$, where X^k denotes the node representation after k messaging passing. Suppose that $\mathbf{R}$ is a full rank matrix with n eigenvalues $\{\lambda_1, \lambda_2, ..., \lambda_n\}$ and n corresponding unit eigenvectors $\{u_1, u_2, ..., u_3\}$. The attribute X can be decomposed as the sum of their projections to these eigenvectors: $X = \sum_{i=1}^{n} x_i u_i$. Then we can represent $\mathbf{R}^k X$ as:

$$\begin{aligned} \mathbf{R}^k X = \mathbf{R}^k \sum_{i=1}^{n} x_i u_i &= \sum_{i=1}^{n} x_i \mathbf{R}^k u_i = \sum_{i=1}^{n} x_i \lambda_i^k u_i \\ &= \lambda_i^k (x_1 u_1 \sum_{i=2}^{x} (\frac{\lambda_i^k}{\lambda_1^k})^2 x_i u_i) \end{aligned} \quad . \tag{12}$$

Here we set $|\lambda_i| > |\lambda_{i+1}|$, and therefore, as k increases, items with smaller magnitude eigenvalues would fade gradually. And the eigenvectors of the surviving items can be regarded as the high-order features of the node. We select eigenvectors with top-k largest eigenvalues as the node embeddings, which can also be adapted to adjacency matrices with negative edges.

Figure 2 illustrates the framework's architecture, consisting of three components: pre-embedding module, GNN module, and prediction module. The pre-embedding module computes eigenvectors with top-k largest eigenvalues (magnitude) and treats the eigenvectors as the pre-embeddings of the node. Then we perform standard normalization on the eigenvectors and normalize the edge weights to the interval [0, 1]. Subsequently, the GNN layers receive the normalized graph with topology-aware node embeddings and further update the node representation with messaging passing. After obtaining the updated node representation, the prediction module predicts an edge's weight by taking the dot product of its connected node representation.

5 DATA

To evaluate our framework we use two different datasets containing trust rating scores from the Bitcoin OTC (over-the-counter) and Bitcoin Alpha platforms [21, 22]. See Table 1 for a summary of the two datasets. We use this to create a signed directed graph with a node representing a Bitcoin user and an edge representing the ratings between two users. The edge weights represent the degree of trust assigned from one user to another, and they are valued from -10 to 10 indicating respectively the highest distrust to the maximum trust level. The significance of these edge weights is discussed in section 4.1. Note, that since the users are anonymous, no additional user features are available.

Table 1: Data summary.

	Bitcoin OTC	Bitcoin Alpha
# of nodes	5,881	3,783
# of edges	35,592	24,186
% of positive edges	89%	93%

6 EXPERIMENTS, RESULTS, AND ANALYSIS

We evaluated our framework with three different variants of GNNs: GCN [20], SGC [49], and GraphSAGE [14].

The propagation rule of GCN is defined by:

$$\mathbf{H}^{l+1} = \sigma(\mathbf{D}^{-\frac{1}{2}}\tilde{\mathbf{A}}\mathbf{D}^{-\frac{1}{2}}\mathbf{H}^l W^l) \, .$$

The propagation rule of SGC is a simplified version of the one used by the GCN that eliminates the activation:

$$\mathbf{H}^{l+1} = \mathbf{D}^{-\frac{1}{2}}\tilde{\mathbf{A}}\mathbf{D}^{-\frac{1}{2}}\mathbf{H}^l W^l \, .$$

For GraphSAGE we use "mean aggregation", and we import edge weights into the aggregation process. For a node i, the GraphSAGE propagation rule is defined by:

$$h_i^{l+1} = \sigma(W^l \cdot \mathrm{MEAN}(e_{ii} \cdot h_i^l \cup e_{ij} \cdot h_j^l, \forall j \in N_i))$$

where e_{ij} denotes the weight of the link from node i to node j.

For all the GNNs, we stack two layers, with the input feature dimension as 64, the hidden feature dimension as 32, and the output feature dimension as 16. The learning rate is 0.001, weight decay

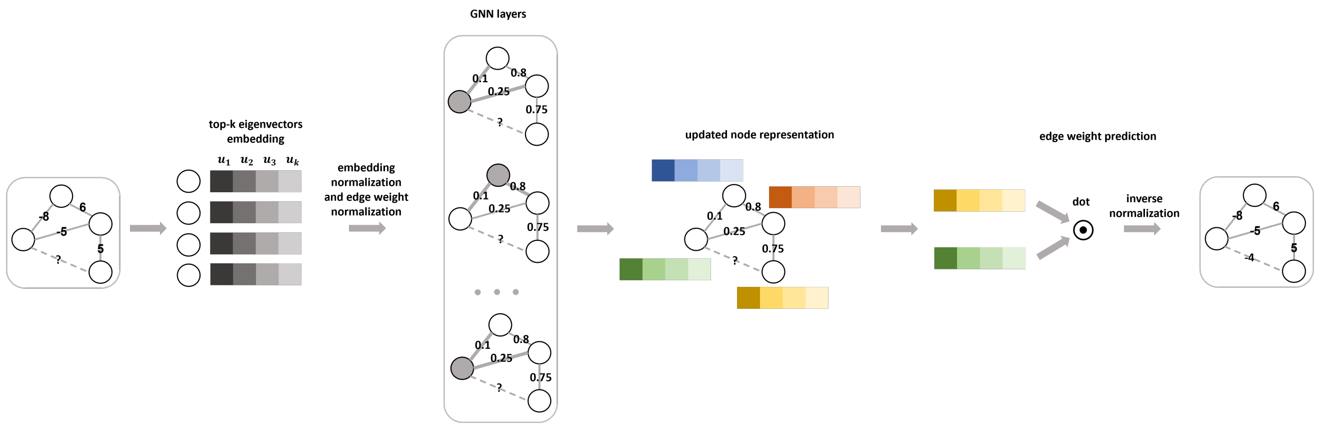
GNN layers
top-k eigenvectors embedding
u_1 u_2 u_3 u_k
embedding normalization and edge weight normalization
updated node representation
edge weight prediction
dot
inverse normalization
-8
6
-5
5
?
0.1
0.8
0.25
0.75
-4

Figure 2: A schematic view of the proposed framework. u_k is the eigenvectors with k-th largest eigenvalues.

is 0.00001. Training epoch is 6000 on bitcoin Alpha and 6500 on bitcoin OTC. For DeepWalk and node2vec, the corpus is constructed by sampling 16 ten-walk-length paths from each node. The window size is 5 and the training epoch is 100. For node2vec, the likelihood of immediately revisiting a node is 0.1 and the control parameter between breadth-first strategy and depth-first strategy is 0.5. The model is trained with MSE loss between the predictions and the actual edge weights.

Through our experiments, we aimed to answer four underlying questions: (1) Does the presence of negative edges degrade GNN performance? (2) Does positive normalization improve embedding quality in GNNs? (3) Which adjacency matrix is optimal for extracting the eigenvectors? (4) Does pre-embedding based on the largest eigenvector-based outperform other embedding methods?

6.1 The Impact of Negative Edges on GNN Performance

To test whether negative edges degrade a GNNs performance, we normalize the edge weights restricted to the interval $[-1, 1]$. Since with the symmetric normalized adjacency matrix, $\mathbf{D}^{-\frac{1}{2}}\tilde{\mathbf{A}}\mathbf{D}^{-\frac{1}{2}}$, the degree cannot be negative, we modify the node degree to be the sum of the magnitude of its connected edge weights such that $d_i = \sum_{j\in N_i} |e_{ij}|$. We then evaluated the influence of negative edges with two kinds of pre-embeddings: random initialization and top-k largest eigenvectors (which we refer to as Eigen-embedding). In Table 2, we observe that in both datasets, all the GNNs exhibit a decrease in predictive accuracy after introducing negative edge weights in neighborhood aggregation. Furthermore, it can be observed that the Eigen-embedding significantly improves the predictive performance for graphs with positive edges. We also observe that in spite of negative edges, in some cases GCNs with Eigen-embedding perform worse than the GCNs with random-initialized embedding. For instance, in Bitcoin Alpha, incorporating Eigen-embeddings increases the test loss of the GCN model from 0.0548 to 0.5665. This result may indicate that negative links introduce noise into message passing, which may ruin the inherent topological information in the eigenvectors. However, overall, our experimental results strongly indicate that negative edges deteriorate message passing in GNNs.

6.2 Positive Normalization and Enhancement of Embedding Quality in GNNs

In the previous section, we demonstrated the negative effect of negative edges on neighborhood aggregation. To mitigate these effects, one approach may be to normalize the edge weights restricted to the interval $[0, 1]$. These negative edges, however, may hold rich semantic information such as “distrust” or “dislike” and mapping them to a positive interval would reduce the power of these semantic signals (i.e., weaken “distrust” to “little trust”), causing information loss. Therefore, it is necessary to investigate whether positive neighborhood aggregation learns adequate topological knowledge and improves prediction performance. Since GNNs consist of neighborhood aggregation and non-linear transformation, we eliminate the neighborhood aggregation module and degenerate the GNNs to a two-layer MLP. According to Table 3, it can be observed that in Bitcoin Alpha, all the GNNs outperform MLP and GraphSAGE performs significantly better than other models. In Bitcoin OTC, with randomly initialized embeddings, all the GNNs perform superior to MLP. However, MLP has made great progress with Eigen-embeddings, overtaking GCN and SGC. GraphSAGE is still the best model. It can be safe to conclude that neighborhood aggregation plays a positive role since all GNNs significantly outperform MLP with randomly initialized embeddings.

Now we explain why eigen-embedding enhanced MLP outperforms GCN and SGC from an intuitive perspective. Both GCN and SGC employ the normalized symmetric adjacency matrices, while GraphSAGE multiplies the positively normalized edge weight with the node representation. Therefore, GCN and SGC perform secondary processing (minimax normalization to [0,1] and symmetric degree normalization) on edge weights. Symmetric degree normalization assumes that the high-degree node would suppress the weight of the edge connecting it. The assumption is valid in the graph with all positive edges. However, when taking negative edges into account, this assumption is questionable, given a high-degree node with nine positive links and one negative link. Intuitively

Table 2: The prediction loss of GNNs with and without negative edges.

Datasets	Bitcoin Alpha			Bitcoin OTC		
Models	**GCN**	**SGC**	**GraphSAGE**	**GCN**	**SGC**	**GraphSAGE**
Positive-random	0.02144 ± 0.00045	0.02238 ± 0.00012	0.01556 ± 0.00027	0.02601 ± 0.00047	0.02904 ± 0.00018	0.02019 ± 0.00020
Negative-random	0.05480 ± 0.00020	0.05519 ± 0.00012	0.03655 ± 0.00028	0.06696 ± 0.00036	0.06856 ± 0.00019	0.04845 ± 0.00065
Positive-eigen	0.01763 ± 0.00017	0.01773 ± 0.00013	0.01129 ± 0.00044	0.01867 ± 0.00019	0.01881 ± 0.00016	0.01390 ± 0.00006
Negative-eigen	0.05665 ± 0.00045	0.05664 ± 0.00037	0.03321 ± 0.00075	0.06191 ± 0.00034	0.06169 ± 0.00014	0.04145 ± 0.00036

the negative link should be paid much attention to (assign higher weight) since it challenges the majority opinions, which indicates the node pointed by the negative link is likely to hold unique properties. However, when performing normalization, all the edge weights are suppressed by the degree. The importance of the only negative link is underestimated.

6.3 Adjacency Matrix Selection for Extracting the Eigenvectors

In Section 3.2, we introduced other two versions of the adjacency matrix: the symmetric normalized adjacency matrix and the random walk adjacency matrix. Each of these is widely used in GNN literature and as such, we used our experiments to assess which one is better suited for eigenvector extraction. The result of these experiments is shown in Table 4, and it can be observed that the performance with eigenvectors extracted from the raw adjacency matrix is associated with significant performance compared to other matrices. This is consistent with our hypothesis in the previous subsection that normalization would ruin negative signals.

6.4 The Effect of Pre-embedding (Based on the Largest Eeigenvector-based) on the Performance

We compared Eigen-embedding from two classical node embedding methods: DeepWalk and node2vec. We sample sixteen ten-length paths from each node using these two methods to generate the corpus and set the window size to five. For node2vec, the trade-off parameter for BFS and DFS is 0.5. A comparison of the performance of these methods is shown in Table 5. Across both the Bitcoin OTC and Alpha datasets, the Eigen-embedding enhanced SGC and GraphSAGE outperform the other methods, whereas Eigen-embedding enhanced GCN performs better than DeepWalk but worse than node2vec. However, given that DeepWalk and node2vec both require expensive network training procedures, Eigen-embedding which does not require training may be considered a better solution in practice.

7 CONCLUSIONS AND FUTURE WORKS

In this work, we explore different approaches of using GNNs to support financial crime detection in networks of bitcoin users. We propose a solution to mitigate the effect of negative edges on the GNNs performance and compensate for the weaknesses of GNNs in learning high-order topological information in modeling the rating network among bitcoin users. Our proposed solution employs the eigenvectors with the top-k largest eigenvalues of the raw adjacency matrix for the pre-embedding of the nodes which are then fed alongside the positively normalized adjacency matrix to the GNN layer to obtain high-quality node embeddings, These embeddings may be used to support a variety of downstream tasks such as fraud detection. Our experiments also illustrate the adverse influence of negative edges in the neighborhood aggregation of GNNs and demonstrate the superiority of GNNs to MLP approaches as well as the superiority of Eigen-embedding. Furthermore, we also investigate the optimal choice of various adjacency matrices and GNNs. This work also provides a constructive reference for future research using advanced graph analysis technology such as GNN to enhance the supervision of financial networks.

REFERENCES

[1] Bithin Alangot, Daniel Reijsbergen, Sarad Venugopalan, and Pawel Szalachowski. 2020. Decentralized lightweight detection of eclipse attacks on bitcoin clients. In *2020 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 337–342.

[2] James Atwood and Don Towsley. 2016. Diffusion-convolutional neural networks. *Advances in neural information processing systems* 29 (2016).

[3] Ui-Jun Baek, Se-Hyun Ji, Jee Tae Park, Min-Seob Lee, Jun-Sang Park, and Myung-Sup Kim. 2019. DDoS attack detection on bitcoin ecosystem using deep-learning. In *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 1–4.

[4] Deng Cai, Xiaofei He, and Jiawei Han. 2007. Spectral regression: A unified subspace learning framework for content-based image retrieval. In *Proceedings of the 15th ACM international conference on Multimedia*. 403–412.

[5] Deepesh Chaudhari, Rachit Agarwal, and Sandeep Kumar Shukla. 2021. Towards Malicious address identification in Bitcoin. In *2021 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 425–432.

[6] Binjie Chen, Fushan Wei, and Chunxiang Gu. 2021. Bitcoin Theft Detection Based on Supervised Machine Learning Algorithms. *Security and Communication Networks* 2021 (2021).

[7] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 04 (Apr. 2020), 3438–3445.

[8] Dawei Cheng, Zhibin Niu, Jie Li, and Changjun Jiang. 2022. Regulating systemic crises: Stemming the contagion risk in networked-loans through deep graph learning. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[9] Zhenjin Dai, Xutao Wang, Pin Ni, Yuming Li, Gangmin Li, and Xuming Bai. 2019. Named entity recognition using BERT BiLSTM CRF for Chinese electronic health records. In *2019 12th international congress on image and signal processing, biomedical engineering and informatics (cisp-bmei)*. IEEE, 1–5.

[10] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 3844–3852.

[11] Joan Bruna Estrach, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral networks and deep locally connected networks on graphs. In *2nd International conference on learning representations, ICLR*, Vol. 2014.

[12] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*. PMLR, 1263–1272.

[13] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[14] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

[15] Jingguang Han, Yuyun Huang, Sha Liu, and Kieran Towey. 2020. Artificial intelligence for anti-money laundering: a review and extension. *Digital Finance*

Table 3: Loss comparison between MLP and GNNs

	Bitcoin Alpha		**Bitcoin OTC**	
	random	**eigen**	**random**	**eigen**
MLP	0.02253 ± 0.00016	0.01920 ± 0.00011	0.03213 ± 0.00026	0.01686 ± 0.00014
GCN	0.02144 ± 0.00045	0.01763 ± 0.00017	0.02601 ± 0.00047	0.01867 ± 0.00019
SGC	0.02238 ± 0.00012	0.01773 ± 0.00013	0.02904 ± 0.00018	0.01881 ± 0.00016
GraphSAGE	0.01556 ± 0.00027	**0.01129 ± 0.00044**	0.02019 ± 0.00020	**0.01390 ± 0.00006**

Table 4: Comparisons between three versions of adjacency matrices.

	Bitcoin Alpha			**Bitcoin OTC**		
	GCN	**SGC**	**GraphSAGE**	**GCN**	**SGC**	**GraphSAGE**
A_{raw}	**0.01763 ± 0.00017**	**0.01773 ± 0.00013**	**0.01129 ± 0.00044**	**0.01867 ± 0.00019**	**0.01881 ± 0.00016**	**0.01390 ± 0.00006**
A_{sym}	0.02527 ± 0.00035	0.02551 ± 0.00005	0.02343 ± 0.00021	0.03320 ± 0.00089	0.03437 ± 0.00036	0.02547 ± 0.00150
A_{rw}	0.02474 ± 0.00013	0.02512 ± 0.00029	0.02119 ± 0.00017	0.03271 ± 0.00118	0.03343 ± 0.00031	0.02598 ± 0.00067

Table 5: Comparisons with other node embedding methods.

Datasets	**Bitcoin Alpha**			**Bitcoin OTC**		
Models	**GCN**	**SGC**	**GraphSAGE**	**GCN**	**SGC**	**GraphSAGE**
DeepWalk	0.01723 ± 0.00022	0.02274 ± 0.00034	0.01750 ± 0.00040	0.02005 ± 0.0005	0.02695 ± 0.00011	0.01935 ± 0.00016
node2vec	**0.01609 ± 0.00017**	0.02136 ± 0.00026	0.01556 ± 0.00050	**0.01765 ± 0.00016**	0.02236 ± 0.00011	0.01709 ± 0.00057
eigen	0.01763 ± 0.00017	**0.01773 ± 0.00013**	**0.01129 ± 0.00044**	0.01867 ± 0.00019	**0.01881 ± 0.00016**	**0.01390 ± 0.00006**

2, 3 (2020), 211–239.

[16] Xiaofei He. 2004. Locality Preserving Projections. In *Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference*, Vol. 16. MIT Press, 153.

[17] Thomas Hofmann and Joachim Buhmann. 1994. Multidimensional scaling and data clustering. In *Proceedings of the 7th International Conference on Neural Information Processing Systems*. 459–466.

[18] Suhwan Ji, Jongmin Kim, and Hyeonseung Im. 2019. A comparative study of bitcoin price prediction using deep learning. *Mathematics* 7, 10 (2019), 898.

[19] Michael Kenning, Jingjing Deng, Michael Edwards, and Xianghua Xie. 2022. A directed graph convolutional neural network for edge-structured signals in link-fault detection. *Pattern Recognition Letters* 153 (2022), 100–106.

[20] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[21] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and VS Subrahmanian. 2018. Rev2: Fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 333–341.

[22] Srijan Kumar, Francesca Spezzano, VS Subrahmanian, and Christos Faloutsos. 2016. Edge weight prediction in weighted signed networks. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 221–230.

[23] Xiaoxiao Li et al. 2020. Explain graph neural networks to understand weighted graph features in node classification. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*. Springer, 57–76.

[24] Yuming Li, Pin Ni, and Victor Chang. 2019. An Empirical Research on the Investment Strategy of Stock Market based on Deep Reinforcement Learning model.. In *COMPLEXIS*. 52–58.

[25] Yuming Li, Pin Ni, and Victor Chang. 2020. Application of deep reinforcement learning in stock trading strategies and stock forecasting. *Computing* 102, 6 (2020), 1305–1322.

[26] Yuming Li, Pin Ni, Junkun Peng, Jiayi Zhu, Zhenjin Dai, Gangmin Li, and Xuming Bai. 2019. A joint model of clinical domain classification and slot filling based on RCNN and BiGRU-CRF. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 6133–6135.

[27] Wai Weng Lo, Siamak Layeghy, and Marius Portmann. 2022. Inspection-L: Practical GNN-Based Money Laundering Detection System for Bitcoin. *arXiv preprint arXiv:2203.10465* (2022).

[28] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[29] Pin Ni, Gangmin Li, Patrick CK Hung, and Victor Chang. 2021. StaResGRU-CNN with CMedLMs: A stacked residual GRU-CNN with pre-trained biomedical language models for predictive intelligence. *Applied Soft Computing* 113 (2021), 107975.

[30] Pin Ni, Yuming Li, and Victor Chang. 2020. Research on text classification based on automatically extracted keywords. *International Journal of Enterprise Information Systems (IJEIS)* 16, 4 (2020), 1–16.

[31] Pin Ni, Yuming Li, and Victor Chang. 2022. Recommendation and sentiment analysis based on consumer review and rating. In *Research Anthology on Implementing Sentiment Analysis Across Multiple Disciplines*. IGI Global, 1633–1649.

[32] Pin Ni, Yuming Li, Gangmin Li, and Victor Chang. 2020. Natural language understanding approaches based on joint task of intent detection and slot filling for IoT voice interaction. *Neural Computing and Applications* 32, 20 (2020), 16149–16166.

[33] Pin Ni, Yuming Li, Gangmin Li, and Victor Chang. 2021. A hybrid siamese neural network for natural language inference in cyber-physical systems. *ACM Transactions on Internet Technology (TOIT)* 21, 2 (2021), 1–25.

[34] Pin Ni, Yuming Li, Jiayi Zhu, Junkun Peng, Zhenjin Dai, Gangmin Li, and Xuming Bai. 2019. Disease diagnosis prediction of emr based on BiGRU-ATT-capsnetwork model. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 6166–6168.

[35] Pin Ni, Ramin Okhrati, Steven Guan, and Victor Chang. 2022. Knowledge Graph and Deep Learning-based Text-to-GQL Model for Intelligent Medical Consultation Chatbot. *Information Systems Frontiers* (2022), 1–19.

[36] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. 2021. Deep learning for anomaly detection: A review. *ACM Computing Surveys (CSUR)* 54, 2 (2021), 1–38.

[37] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[38] Ziyue Qiu, Yuming Li, Pin Ni, and Gangmin Li. 2019. Credit risk scoring analysis based on machine learning models. In *2019 6th International Conference on Information Science and Control Engineering (ICISCE)*. IEEE, 220–224.

[39] Zhengxiang Shi, Yue Feng, and Aldo Lipani. 2022. Learning to Execute Actions or Ask Clarification Questions. In *Findings of the Association for Computational Linguistics: NAACL 2022*. 2060–2070.

[40] Zhengxiang Shi, Pin Ni, To Eun Kim, Meihui Wang, and Aldo Lipani. 2022. Attention-based Ingredient Phrase Parser. In *European Symposium on Artificial Neural Networks, Computational Intelligence*.

[41] Zhengxiang Shi, Qiang Zhang, and Aldo Lipani. 2022. Stepgame: A new benchmark for robust multi-hop spatial reasoning in texts. *Association for the Advancement of Artificial Intelligence* (2022).

[42] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*. 1067–1077.

[43] Bishenghui Tao, Hong-Ning Dai, Jiajing Wu, Ivan Wang-Hei Ho, Zibin Zheng, and Chak Fong Cheang. 2021. Complex network analysis of the bitcoin transaction

network. *IEEE Transactions on Circuits and Systems II: Express Briefs* 69, 3 (2021), 1009–1013.

[44] Hao Tian, Yang Li, Yue Cai, Xiaohong Shi, and Zibin Zheng. 2021. Attention-Based Graph Neural Network for Identifying Illicit Bitcoin Addresses. In *International Conference on Blockchain and Trustworthy Systems*. Springer, 147–162.

[45] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[46] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[47] Mark Weber, Jie Chen, Toyotaro Suzumura, Aldo Pareja, Tengfei Ma, Hiroki Kanezashi, Tim Kaler, Charles E Leiserson, and Tao B Schardl. 2018. Scalable graph learning for anti-money laundering: A first look. *arXiv preprint arXiv:1812.00076* (2018).

[48] Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I Weidele, Claudio Bellei, Tom Robinson, and Charles E Leiserson. 2019. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591* (2019).

[49] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. PMLR, 6861–6871.

[50] Yan Wu, Fang Tao, Lu Liu, Jiayan Gu, John Panneerselvam, Rongbo Zhu, and Mohammad Nasir Shahzad. 2020. A bitcoin transaction network analytic method for future blockchain forensic investigation. *IEEE Transactions on Network Science and Engineering* 8, 2 (2020), 1230–1241.

[51] Xiaohan Xing, Fan Yang, Hang Li, Jun Zhang, Yu Zhao, Mingxuan Gao, Junzhou Huang, and Jianhua Yao. 2022. Multi-level attention graph neural network based on co-expression gene modules for disease diagnosis and prognosis. *Bioinformatics* 38, 8 (2022), 2178–2186.

[52] QiAo Yuan, Pin Ni, Junru Liu, Xiangzhi Tong, Hanzhe Lu, Gangmin Li, and Steven Guan. 2021. An Encoder-decoder Architecture with Graph Convolutional Networks for Abstractive Summarization. In *2021 IEEE 4th International Conference on Big Data and Artificial Intelligence (BDAI)*. IEEE, 91–97.

[53] Yingxue Zhang, Florence Regol, Soumyasundar Pal, Sakif Khan, Liheng Ma, and Mark Coates. 2021. Detection and defense of topological adversarial attacks on graphs. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2989–2997.

[54] Baokun Zheng, Liehuang Zhu, Meng Shen, Xiaojiang Du, Jing Yang, Feng Gao, Yandong Li, Chuan Zhang, Sheng Liu, and Shu Yin. 2017. Malicious bitcoin transaction tracing using incidence relation clustering. In *International Conference on Mobile Networks and Management*. Springer, 313–323.

[55] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81.

[56] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems* 33 (2020), 7793–7804.