Journal Pre-proof

Adaptive multi-channel Bayesian graph attention network for IoT transaction security

Zhaowei Liu, Dong Yang, Shenqiang Wang, Hang Su

PII: S2352-8648(22)00259-0

DOI: https://doi.org/10.1016/j.dcan.2022.11.018

Reference: DCAN 573

To appear in: *Digital Communications and Networks*

Received Date: 20 September 2022

Revised Date: 24 October 2022

Accepted Date: 29 November 2022

Please cite this article as: Z. Liu, D. Yang, S. Wang, H. Su, Adaptive multi-channel Bayesian graph attention network for IoT transaction security, *Digital Communications and Networks* (2023), doi: https://doi.org/10.1016/j.dcan.2022.11.018.

Digital Communications and Networks(DCN)

HOSTED BY

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/dcan

Adaptive Multi-channel Bayesian Graph Attention Network for IoT Transaction Security

Zhaowei Liu*[a], Dong Yang[a], Shenqiang Wang[a], Hang Su[b]

[a]School of Computer and Control Engineering, Yantai University, Yantai 264005, China
[b]Department of Electronics, Information and Bioengineering, Politecnico di Milano, Milan 20133, Italy

Abstract

With the rapid advancement of 5G technology, the Internet of Things (IoT) has entered a new phase of application and is rapidly becoming a significant force in promoting economic development. Due to the vast amounts of data created by numerous 5G IoT devices, the Ethereum platform has become a tool for the storage and sharing of IoT device data, thanks to its open and tamper-resistant characteristics. So, Ethereum account security is necessary for the Internet of Things to grow quickly and improve people's lives. By modeling Ethereum transaction records as a transaction network, the account types are well identified by the Ethereum account classification system established based on Graph Neural Networks (GNNs). This work first investigates the Ethereum transaction network, Surprisingly, experimental metrics reveal that the Ethereum transaction network is neither optimal nor even satisfactory in terms of accurately representing transactions per account. This flaw may significantly impede the classification capability of GNNs, which is mostly governed by their attributes. This work proposes an Adaptive Multi-channel Bayesian Graph Attention Network (AMBGAT) for Ethereum account classification to address this difficulty. AMBGAT uses attention to enhance node features, estimate graph topology structure that conforms to the ground truth, and efficiently extract node features pertinent to downstream tasks. An extensive experiment with actual Ethereum transaction data demonstrates that AMBGAT obtains competitive performance in the classification of Ethereum accounts while accurately anticipating the graph's topology.

KEYWORDS:
Internet of Things, Graph Representation Learning, Node classification, Security Mechanism.

1. Introduction

The Internet of Things (IoT) [1] is a network of interconnected items that facilitates content retrieval and service discovery. With the complete implementation of 5G technology, the Internet of Things data volume is expanding at an unprecedented rate. With the help of a large amount of IoT data, data mining technology has played an important role in science and technology, economics, energy, smart cities, and other fields. In addition, as artificial intelligence technology advances, the potential usefulness of IoT data will be investigated further. Consequently, the transactional value of IoT data has been acknowledged by an increasing number of individuals and organizations. The emergence of the IoT data transaction platform is a response to the growing need for IoT transactions. Currently, the majority of the most popular IoT data transaction platforms store and exchange data via a reputable third party [2]. This has a lot of problems, such as ignoring data security, which could lead to private information being shared [3], and relying on a third-party platform, which will add to administrative costs and charge too much for data transactions.

With the development of Blockchain [4], trustworthy intermediary transactions can be conducted in an

*Zhaowei Liu (Corresponding author) (email:lzw@ytu.edu.cn).
[1]Dong Yang (email:yangdong@s.ytu.edu.cn).
[2]Shenqiang Wang(email:wsqaahh@foxmail.com).
[3]Hang Su(email:hang.su@polimi.it).

anonymous setting, eliminating the requirement for trusted third parties. All transaction records on the blockchain are immutable and publicly visible, and they are maintained via peer-to-peer networks employing novel consensus mechanisms. Therefore, an increasing number of Internet of Things researchers are utilizing blockchain technology to achieve point-to-point value transfer across unknown nodes [5, 6, 7]. This opportunity has been seized by the new, trustworthy transaction platform, the Ethereum transaction platform [8]. The Ethereum platform provides a universal computing and Turing-complete language to allow smart contracts, which significantly reduces the difficulty of implementing blockchain technology. In order to promote the use of smart contracts, the Ethereum network developed the concept of an account, which officially existed as an address. In the sphere of the Internet of Things, more and more researchers are utilizing Ethereum to facilitate transactions between untrusted nodes. Yutaka et al.[9] utilize the Ethereum platform to store IoT data and conduct peer-to-peer transactions. Raj et al.[10] propose implementing smart contracts on a private Ethereum Blockchain to enhance the security of IoT data. Mehedi et al.[11] build the Ethereum Blockchain system to provide access control to key resources for the IoT. Sun et al.[12] use Ethereum to construct a reliable IoT system and to propose a refueling mechanism for IoT devices. In other words, the Ethereum platform has become a crucial platform for the secure storage and exchange of IoT data.

Blockchain's anonymity is also a double-edged sword, as everything has two sides. Rampant cybercriminals, particularly on the Ethereum platform, use the anonymity feature to commit a variety of cybercrimes [13], posing significant risks to legitimate users and jeopardizing societal security. Numerous financial crimes have been recorded on the Ethereum platform in recent years, and more than 10% of Ethereum accounts have been compromised in various ways, including phishing [14], honeypot [15], money laundering [16], Ponzi schemes [17], and so on. Specifically, phishing scams and Ponzi schemes account for the vast majority of Ethereum-related crimes, demonstrating that transaction security has become a key concern in the Ethereum ecosystem. In order to ensure the creation of a safe and harmonious community, it is necessary to differentiate between different types of anonymous accounts in Ethereum. This will help identify abnormal transaction behaviors, provide more hints for transaction tracking, improve the auditability of the entire platform, and ultimately provide a guarantee for safe storage and transaction of IoT data.

The account-centric transaction model of the Ethereum platform [18] allows each transaction to be viewed as a trusted operation from one account to another, like transfers and bills. The Ethereum account classification issue can be turned into the node classification problem in the Ethereum transaction network by representing Ethereum transaction data as the Ethereum transaction network. Farrugia et al.[19] build an Ethernet transaction network containing both genuine and fraudulent accounts and propose a classifier for recognizing fraudulent accounts in an Ethernet transaction network. Hu et al.[20] offer a data slicing-based methodology to establish an Ethernet smart contract network with features and a transaction-based classification and detection approach for Ethernet smart contracts in order to classify various sorts of accounts in the Ethernet transaction network. Many researchers rely on network embedding techniques to capture the interaction characteristics of accounts in a network since machine learning algorithms excel at processing well-defined and fixed-length inputs and outputs. Chen et al.[21] suggest a graph-based cascading feature extraction method for retrieving rich information about transaction structures. Wang et al.[22] design a heterogeneous Ethernet transaction network and propose a network embedding algorithm based on the heterogeneous network to mine the implicit information in Ethernet transactions. GNNs are also an excellent model for handling blockchain transaction networks.Chen et al.[14] first used the GCN [23] method for blockchain transaction networks to identify phishing accounts. However, they did not update the classic GCN model to match the blockchain transaction network's peculiarities.

As an end-to-end learning framework, are GNN models adequate for tackling the Ethereum transaction network account classification task? This is a critical question since transaction networks possess some special characteristics. A rational response to this issue can assist us in gaining a fundamental knowledge of the capabilities and limitations of GNNs on the Ethereum transaction network. This immediately inspires our research.

This work analyzes key Ethereum transaction network characteristics as the first contribution of this research. Not only does our research aid in comprehending the distinctive properties of the Ethereum transaction network, but the network analysis results reveal that the Ethereum transaction network has low homogeneity (i.e., the proportion of a node's neighbors that typically belong to the same class), which makes it difficult for most GNNs to be directly generalized to the Ethereum transaction network. In the current payment system, there exist transaction linkages between accounts of two distinct sorts. For instance, exchange hot wallet accounts typically perform transactions with many services and users, causing the original transaction network's topology to not be qualitatively ideal. The inability to learn the most relevant node embeddings for downstream tasks based on the characteristics of the Ethereum transaction network may severely hamper the account categorization capability of GNNs due to this weakness.

Once the weaknesses of good GNNs on the Ethernet trading network were identified, the question arose naturally: "Can we remedy this weakness while designing a new GNN that preserves the advantages of neighborhood aggregation of GNNs while simultaneously estimating the ground truth network topology to improve the node embedding capability of GNN?"

This work addresses this challenge and proposes a classification model for Ethereum accounts. The main idea behind the model is to first estimate the network topology of the ground truth using the original trading network and node features, then use multiple attention convolution modules to learn the node embedding information from the original graph and the estimated graph , and then adaptively combine the information most relevant to the account classification task from the two views to improve GNN's ability to embed nodes.

Technically, for the aggregation of node neighborhood features, the importance of neighborhood features is calculated by the convolution module with an attention mechanism, which enables the creation of more precise node embedding information. For the production of the estimation graph, the stochastic block model (SBM) [24] is used to reflect the underlying structural generation of the estimation graph based on the neighborhood aggregation property of GNNs in order to provide it with community structure attributes. Multiple kinds of information are retrieved simultaneously as observations of the estimate graph in order to minimise its bias, and the posterior distribution of the graph structure is finally inferred using Bayesian inference. The node features are propagated across both the estimated graph and the original graph, thereby extracting the specific embeddings of each node in both channels. In the meantime, the common features in both channels are retrieved via a shared-parameter convolution module. An attention mechanism automatically learns the important weights of the two embeddings of the original and estimated graphs in order to adaptively fuse them. Through iterative optimization, the estimate graph, adaptive weights, and model parameters are ultimately mutually enhanced.

In summary, we summarize three contributions:

1. As for the transaction security of the Internet of Things, this work is the first one based on Bayesian inference to estimate the graph structure to adapt to the GNN learning mechanism, and feature extraction is carried out on both the original graph and the estimated graph to give the most favorable embedding for downstream tasks.
2. This work proposes a new GNN model for the Ethereum account classification task that generates meaningful graph structures by calculating the posterior distribution of the original graph structure via Bayesian inference. Then, the attention mechanism is then used to show how important the original graph and the estimated graph are to the Ethereum account classification task. Finally, perform iterative optimization of the estimated graph, adaptive weights, and model parameters.
3. Through a large number of experiments on the challenging Ethereum transaction data set, AMBGAT is demonstrated to have the expected performance. In addition, AMBGAT is also analyzed for other meaningful properties.

The remaining sections are organized as follows. In Section 2, some related work is reviewed. The characteristics of the Ethereum transaction network are examined in Section 3. In Section 4, the proposed AMBGAT model is qualitatively analyzed. In Section 5, the proposed model is quantitatively analyzed by experiments, and this work is concluded in Section 6.

2. Related work

In line with our research focus, the most relevant work on both graph neural networks and graph topology learning is briefly reviewed, and the feature engineering-based approach to classifying Ethernet accounts is summarized.

2.1. Graph Neural Networks

Most of the current GNNs can be classified into five categories, namely Graph Convolutional Network (GCN), Graph Auto-encoder (GAE), Graph Generative Network (GGN), Graph Recurrent Network (GRN), and Graph Attention Network (GAT). There are two types of convolution operations performed by graph convolution networks: one is graph convolution based on spectral decomposition, and the other is spatial graph convolution based on spatial transformation. Specifically, [25] utilises the eigenvectors of the Laplacian matrix of the graph to perform the graph convolution operation in the Fourier domain. [26] applies Chebyshev polynomials to solve for the eigenmatrices to improve efficiency. [23] proposes a semi-supervised GCN node classification model and further improves the accuracy and learning ability by deepening the depth of the network and reducing the neighbourhood width. Spatial graph convolution learns the representation of each node from the spatial features of the graph data structure. [27] performs inductive graph convolution by aggregating the neighbourhood features generated from multiple iterations. [28] interprets the graph convolution as an integral transformation of the embedding function under the probability measure based on node importance sampling, resulting in improved prediction accuracy. Graph Auto-encoder (GAE) is an artificial neural network that learns the representation of the input information on graph data, including two parts: an encoder and a decoder. [29] proposes a Marginalized Graph Autoencoder (MGAE) algorithm. It uses a graph convolutional network layer based on spectral decomposition

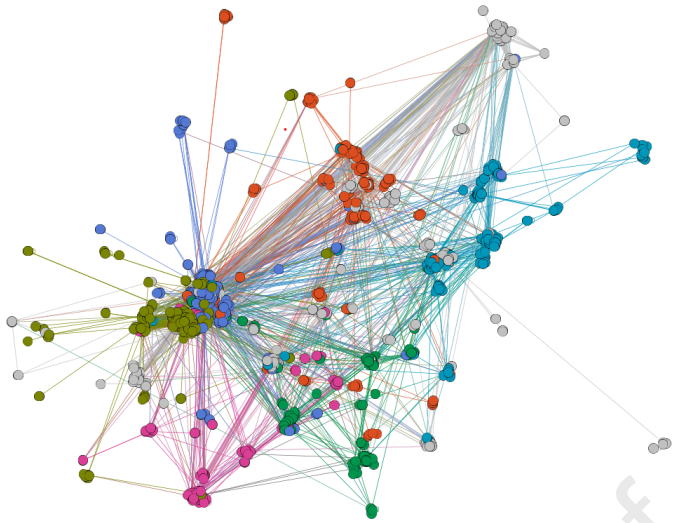


Fig. 1. A visualization of the network structure of sampled transactions, where color indicates the class of the node.

in the autoencoder to integrate node attribute features and graph structure information, enabling data interaction between them. [30] proposes the Variational Graph Auto-encoder (VGAE), which enables the encoder to learn the distribution of a low-dimensional vector representation, constraining the intermediate hidden variables in a distribution. A Graph Generative Network (GGN) is a class of GNN used to generate graph data. [31] proposes the Graphical Generative Adversarial Network (Graphical-GAN), which learns graph structure using a Bayesian network. Learning is performed, and the generator and trainer are jointly trained by an expectation propagation algorithm. The literature [32] proposes a Graph Topology Interpolator (GTI) based on GAN, which reconstructs the adjacency matrix based on edge weights. Graph Recurrent Network (GRN) usually converts graph data into sequences, which evolve and change recursively during the training process. In [33], a Gated Graph Sequence Neural Network (GGS-NN) is proposed to generate a vector of implicit representations by encoding graph sequence features. The literature [34] proposes a Spatio-temporal Attentive Recurrent Network (STAR) capable of learning graph spatio-temporal contextual information and extracting neighbourhood vector representations by sampling and aggregating local neighbourhood nodes. The graph attention network allows the graph neural network to focus only on the information needed for task learning by introducing an attention mechanism. The literature [35] assigns a weight factor to each neighbor based on the characteristics of the neighbouring nodes when aggregating them. The literature [36] provides an unsupervised self-attentive mechanism that can be used for inductive learning. [37] proposes a Graph Attention Model for Multi-Label Learning (GAML), which improves multi-label classification performance and interpretability.

Currently, the majority of graph neural networks assume that the original graph is true, which severely limits their performance in downstream tasks.

2.2. *Graph topology learning*

Graph structure learning has been the subject of several studies in graph neural networks [38] [39] [40]. In addition to these studies, a new graph structure for learning has recently been proposed on undirected graphs. Our work studies graph topology learning on undirected graphs by inscribing Ethernet transaction data as undirected graph transaction networks. The literature [41] proposes a minimally sufficient structural model of the optimal graph structure. The literature [42] obtains the ideal graph structure by adding some constraints, such as low rank and feature smoothing. On other graph types, literature [43] proposes a framework for optimal graph structure learning on heterogeneous graphs. Literature [44] provides a GaN-based directed graph generation method to generate source and target nodes of nodes by joint learning. However, these methods do not provide a framework for Ethernet trading networks to determine which is the "optimal" structure for the downstream task of improving GNNs between the learned graph structure and the original graph structure.

2.3. *Ethereum account classification method*

Malicious accounts on the Ethereum platform pose a great threat to the health of the Ethereum trading ecosystem. As a kind of structured data, graphs can explain the precise relationships between data elements, so many scholars have modeled Ethernet transaction data as graph structures to implement Ethernet account detection and classification [22], i.e., accounts and transactions are represented as nodes and connected edges in a graph, respectively. [45] proposes a graph learning method based on random wandering for feature extraction of phishing users on Ethernet by learning the structural homogeneity and monetary homogeneity of the Ethernet transaction network. [14]

Table 1
Notations and Explanations.

Notations	Explanations
G	Graph depicted by Ethereum transaction data
A	Symmetric adjacency matrix
X	Node feature matrix
V	Nodes in the graph
Y	Labels of the nodes
h	Initial node features
h'	Node features after mapping
Z	The embedding of nodes
W	Model parameters that can be learned
ϑ	The convolution module parameters
S	The generated graph by the SBM model
Q	The estimated graph by the Bayesian inference

Table 2
Statistical information about the Ethereum dataset.

Nodes	Edges	Classes	Features	Labeled nodes
1,124,130	3,752,659	7	22	3,785

proposes a detection method based on graph convolutional networks and auto-encoders to distinguish network phishing accounts by representing GCNs as encoders and their output inner products as decoders to approximate the adjacency matrix to obtain a representation of nodes. [46] applies deep learning to a time series prediction task for the ethereum cryptocurrency, demonstrating the effectiveness of deep learning in predicting the value of transactions. [47] proposes a filtering-enhanced graph neural network to solve the account classification task in Ether, aggregating neighbors of importance by filtering components and using higher-order neighborhood information to enhance the node representation. [48] uses the network embedding method node2ves to extract latent features of accounts, followed by support vector machines for account classification. [49] further enriches the work of [48] by proposing a network embedding algorithm that considers timestamps and transaction volumes to extract node features in the network. [50] proposes a graph-based cascading feature extraction method to extract transaction structure information and a double sampling integration algorithm to classify data with category imbalance. [51] proposes a feature engineering-based phishing anomaly detection framework and provides a robustness testing algorithm for the phishing detection framework. [52] uses an improved graph embedding method to extract latent features of subgraphs, followed by a support vector machine to identify network fraud nodes.

In summary, the current approaches based on feature engineering to achieve Ethereum account classification mainly include network embedding methods represented by random wandering [53] [54] and deep learning methods represented by graph neural networks [23]. However, there is little work to consider the homogeneity of graphs and re-estimate the graph structure to improve the performance of the model.

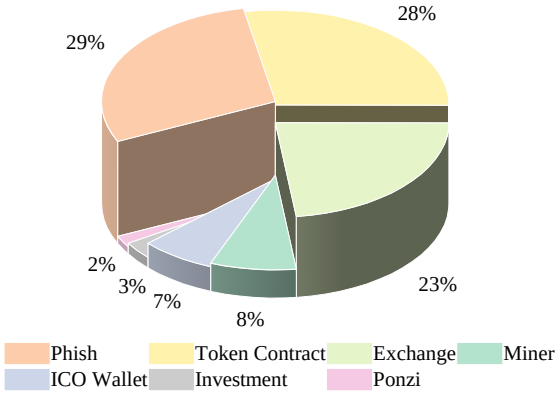


Fig. 2. Our collection of seven typical ethereum accounts.

Table 3
Network metrics for the Ethereum dataset.

Average degree	Average path length	Clustering coefficient	Homophily ratio
6.042	3.844	0.192	0.367

3. Problem Statement and Network Analysis

This section begins with a brief description of the proposed problem, followed by information regarding the source and collection of the Ethereum transaction network dataset. Following this, the characteristics of the Ethereum transaction network are elaborated and compared to those of other networks. Finally, design considerations for a new GNN based on the distinct characteristics of the Ethereum transaction network are presented.

3.1. Problem Statement

Ethereum transaction data can be depicted as graph $G = (A, X)$, where $A \in \mathbb{R}^{n\times n}$ is the symmetric adjacency matrix of n nodes and $X = \left[x_1, x_2, ..., x_N\right] \in \mathbb{R}^{N\times D}$ is the node feature matrix of d dimensions. Specifically, each node represents the corresponding Ethereum account, and the adjacency matrix defines the Ethereum account's transaction relationship. A_{ij} equal to 1 or 0 indicates that there is a transaction relationship or no transaction relationship between nodes i and j, respectively. We concentrate on the semi-supervised node classification task in which only a small percentage of nodes $V_L = \left\{v_1, v_2, ..., v_l\right\}$ are associated with matching labels $Y_L = \left\{y_1, y_2, ..., y_l\right\}$, where v_i's label is y_i, and we assume that each node's label has C categories. Table 1 gives some notations and explanations used throughout this paper.

3.2. Dataset and Network Analysis

Access to cryptocurrency transaction data is made easier by the blockchain's transparency. In this

Table 4
Statistics of homophily ratios for different networks.

	Ethereum	Citeseer[1]	Pubmed[1]	BlogCatalog[1]	Flickr[1]
Original graph	0.367	0.67	0.79	0.39	0.23
Feature graph[2]	0.56±0.01	0.60±0.036	0.74±0.012	0.83±0.027	0.81±0.035

[1] Citeseer, pubmed [23] are citation networks, and BlogCatalog, Flickr [55] are social networks.
[2] The k-NN algorithm is used to generate the feature graph, and the outcome is the mean and standard deviation of k from 2 to 9.

work, the well-known Etherscan is used to collect Ethereum account tags and transaction details. Following the collection methodology of previous researchers [56], we first obtained 3785 account details with ground truth labels, then collected the first-order neighborhood of each account and randomly sampled some second-order neighborhoods, and lastly removed transactions with ambiguous addresses and duplicates. The final dataset consists of an undirected Ethereum transaction network with 1,124,130 nodes and 3,752,659 edges, as depicted in Table 2 and Figure 1. For each node in the Ethereum transaction network, 22 distinguishing features are extracted, including account balance, transaction frequency, transaction amount, and transaction data. In the nodes with real labels, there are a total of 7 different categories, namely: ICO Wallet, Token Contract, Exchange, Miner, Investment, Phish/Hack, and Ponzi, as shown in Figure 2.

Three of the most potent network topology metrics of the Ethereum transaction network and one indicator relating to GNN properties are analyzed in order to design a graph embedding method appropriate for the Ethereum transaction network. The average degree metrics, clustering coefficients, average path lengths, and homophily [57] of the Ethereum dataset are analyzed, as demonstrated in Table 3. The average degree of all nodes in the proposed Ethereum dataset is 6.042, and the degree distribution follows a power-law distribution [58], showing that the majority of nodes have low degrees. This network has a clustering coefficient of 0.192, indicating that there are very few transaction linkages between adjacent accounts of an account. The average path length of this network is 3.844, indicating that the network is comprised of small worlds and that each account is connected to others over relatively short paths. The maximum value of the homogeneity coefficient, which measures whether a network node likes to connect with other nodes with the same label, is 1. This network's homogeneity coefficient is 0.367, indicating that most nodes with the same label do not tend to be connected to one another, whereas the performance of GNNs may be significantly impaired on graphs with poor homogeneity [38]. Since this result is incongruous with the nature of GNNs, we question the precision or optimality of GNNs applied to the Ethereum transaction network.

Furthermore, in order to examine the topological structure of the original Ethereum transaction dataset, this section compares it with social networks and citation networks, and Table 4 displays the homogeneity ratio coefficients of the original and feature graphs in various networks. As can be seen, the homogeneity of the Ethereum transaction network is far lower than that of the citation network, and the feature graphs derived from the node features are likewise significantly lower than those of the four other common networks. In terms of node features, this indicates that interconnected nodes in the Ethereum transaction network have different features.

In conclusion, the Ethereum transaction network features certain topological characteristics, including a low average degree metric, a very low clustering coefficient, and a short average path length. Notably, the Ethereum transaction network has a smaller homogeneity ratio and a greater feature distance than other types of networks, indicating that labels and feature information between accounts with transaction interactions in the network may not be precisely the same. This metric suggests that it is difficult to directly generalize the majority of GNNs to the Ethereum transaction network due to its heterogeneity. If the original Ethereum transaction network is used directly to aggregate the feature information from the local neighborhood, the feature information of the central node could become confused by the feature information of nodes from different labels. Consequently, the analytical results inspire us to estimate a new graph structure for efficiently learning node embedding information.

4. THE PROPOSED MODEL

In this section, the proposed adaptive Multi-channel Bayesian graph attention network , which can learn the most pertinent feature information for downstream tasks in networks with low homogeneity, is discussed. Figure 3 displays the overall framework. The proposed model first employs a convolution module with an attention mechanism to compute the significance of neighborhood features, thus producing more accurate node embedding data. Then, the proposed model estimates a novel graph structure using Bayesian inference to address the poor homogeneity issue of the Ethereum transaction network. Next, the proposed

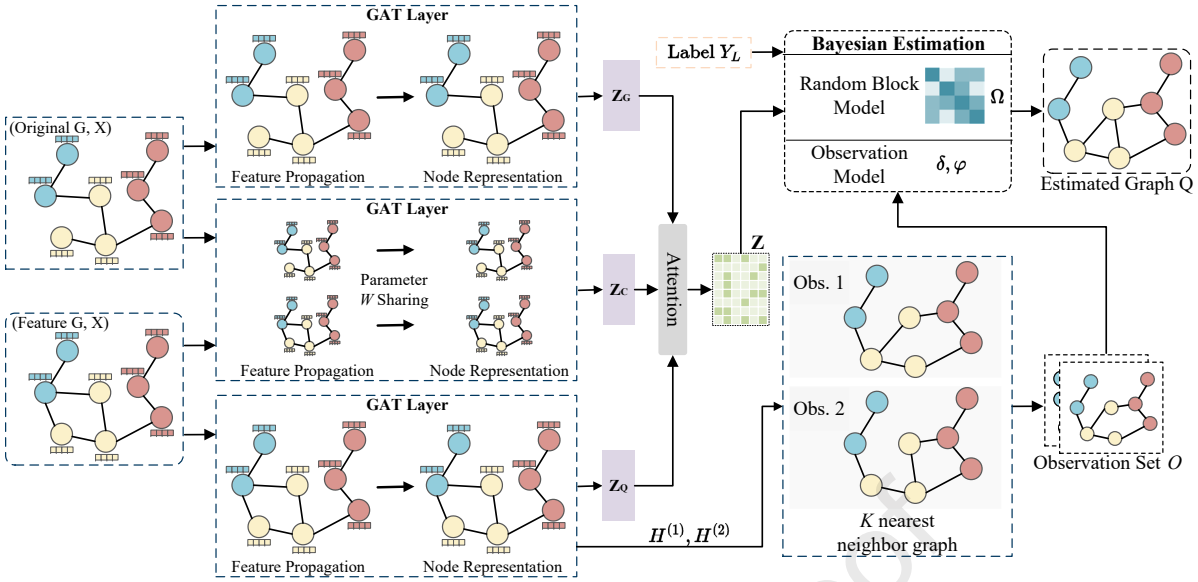


Fig. 3. The general framework of our proposed AMBGAT.

model utilises an adaptive fusion technique for mining multifaceted node feature information on two graph structures to create the most relevant node features for subsequent tasks. Finally, the process of joint iterative optimization of graph structure estimation, attention learning, and GNN parameter learning is illustrated.

4.1. Attentional convolution module

This subsection introduces the attention-based convolution module, the key module for learning node embedding information. Contrary to the conventional neighborhood feature aggregation model GCN [23], the attention-based convolution module is able to assign different importance to different nodes in the neighborhood when learning different numbers of neighborhood features of the target node without requiring prior knowledge of the entire graph structure.

As the backbone of an attention-based neighborhood feature aggregation model, the graph attention network (GAT) [35] is typically selected. Given a collection of node features $\left\{h_i^l\right\}_{i=1,\ldots,N}$, each $h_i^l \in \mathbb{R}^{N\times F}$, where N is the number of nodes and F is the number of node features, the attentional convolution module translates them to the following layer, denoted by $\left\{h_i'\right\}_{i=1,\ldots,N}$. To describe the correlation between neighboring nodes v_i and v_j, the proposed model uses the normalized attentional mechanism of the GAT model, as shown below:

$$f_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\tilde{\mathrm{a}}^T\left(\mathrm{W}^l\,\mathrm{h}_i^l \| \mathrm{W}^l\,\mathrm{h}_j^l\right)\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\tilde{\mathrm{a}}^T\left(\mathrm{W}^l\,\mathrm{h}_i^l \| \mathrm{W}^l\,\mathrm{h}_k^l\right)\right)\right)}, \tag{1}$$

where $\tilde{\mathrm{a}} \in \mathbb{R}^{2F'}$ is the parameter vector of the feed-forward layer, $\|$ represents the tandem operation, $W \in \mathbb{R}^{F'\times F}$ is the weight matrix applied to the linear transformation of each node, and $\mathcal{N}_i$ is the first-order neighborhood of node i in the graph.

To stabilize the self-attentive learning procedure utilizing K independent attention coefficients, the aggregation rule of the convolution module in the hidden layer is designated as follows:

$$h_i' = \|_{k=1}^{K}\sigma\left(\sum_{j\in\mathcal{N}_i} f_{ij}^k \mathbf{W}^k h_j\right), \tag{2}$$

where $\|$ represents tandem, f_{ij}^k is the attention coefficient normalized by the k-th, W^k is the trainable weight matrix, and σ represents the nonlinear activation function.

On the basis of the GAT aggregation technique, the proposed model develops two convolution modules: the Personality Convolution Module and the Common Convolution Module.

4.1.1. Personality Convolution Module

The personality convolution module can learn information from the original and estimated topological graphs, respectively. Inputting the original topological graph G, the predicted value of node i at the last (prediction) layer is stated as:

$$z_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i} f_{ij}^k \mathbf{W}^k h_j\right), \tag{3}$$

where f_{ij}^k is the attention coefficient normalized by the k-th, W^k is the trainable weight matrix in the personality convolution module, and σ is the nonlinear activation function. The output features of the last layer of the original topological graph G are expressed as Z_G, such that individualized information Z_G can be learned from the original topological graph G. To discover the personalized information Z_Q in the estimated graph Q, the estimated graph Q is entered and the prediction information Z_Q is calculated in the same way as for the original topological graph.

4.1.2. Common Convolution Module

The common convolution module is able to extract the common data from both graphs. It is difficult to predict which portion of the original topological graph G, the estimated graph Q, or the common portion of both is most relevant to the downstream task. Consequently, it is required to not only learn the personality information in each graph structure but also introduce the common convolution module. By gaining a deeper understanding of the common features, it is possible to determine with greater flexibility which aspect of the features is the most essential. On the original topological graph G, the predicted value of the prediction (final) layer of node i in the common convolution module can be expressed as:

$$z_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i} f_{ij}^{k}\mathrm{W}_c^{k}h_j\right), \tag{4}$$

where f_{ij}^{k} is the attention coefficient normalized by k-th, W_c^k is the trainable weight matrix in the common convolution module, and σ is the nonlinear activation function. From the initial topological graph G, the features Z_{CG} are derived. The weight matrix W_c is shared between the two graph structures in order to extract the common information. The features Z_{CQ} of the estimated graph Q are learned in the public convolution module in the same manner as those on the original topological graph G, and the predicted features Z_c on both graph structures are indicated as follows:

$$Z_c = \left(Z_{CG} + Z_{CQ}\right)/2. \tag{5}$$

In this section, the developed convolution module can learn two personality features, Z_G and Z_Q, as well as a single common feature, Z_C. The method for computing the correlation between each feature and the node labels is provided in the following section.

4.2. Adaptive Mechanism

The final node features are produced using an adaptive node feature fusion technique based on an attention mechanism that considers which of the feature information Z_G, feature information Z_Q, and feature information Z_C is most important for improving the downstream tasks of GNN.

For the feature information $[Z_G, Z_Q, Z_C]$, learn the attention coefficients $[\alpha_G, \alpha_Q, \alpha_C] \in \mathbb{R}^{n\times 1}$ corresponding to it. $Z_G^i \in \mathbb{R}^{h\times 1}$ represents the feature vector of the node i in Z_G. First apply a nonlinear transform to the feature vector Z_G^i and then multiply it by a shared attention vector $\omega \in \mathbb{R}^{h'\times 1}$ to determine its attention value α_G^i, which is written as:

$$\alpha_G^i = \omega^T \cdot \tanh\left(W \cdot \left(z_T^i\right)^T + b\right), \tag{6}$$

where, $W \in \mathbb{R}^{h'\times h}$ represents the weight matrix, while $b\omega \in \mathbb{R}^{h'\times 1}$ represents the bias vector. To determine the attention values α_Q^i and α_C^i for Z_Q and Z_C, utilize the same method. Normalize the attention values $[\alpha_G^i, \alpha_Q^i, \alpha_C^i]$ with the softmax method to produce the final weight coefficients a_G^i associated with the labels as follows:

$$a_G^i = softmax\left(\alpha_G^i\right) = \frac{\exp\left(\alpha_G^i\right)}{\exp\left(\alpha_G^i\right) + \exp\left(\alpha_Q^i\right) + \exp\left(\alpha_C^i\right)}. \tag{7}$$

Similarly, $a_Q^i = softmax\left(\alpha_Q^i\right)$ and $a_C^i = softmax\left(\alpha_C^i\right)$. Consequently, for all n nodes, the corresponding weight coefficients in the feature information Z_G, feature information Z_Q, and feature information Z_C are $a_G = \mathrm{diag}\left(\left[a_G^i\right]\right)$, $a_Q = \mathrm{diag}\left(\left[a_Q^i\right]\right)$, and $a_C = \mathrm{diag}\left(\left[a_C^i\right]\right) \in \mathbb{R}^{n\times 1}$, respectively. These three weight coefficients are then combined with the corresponding feature information to produce the final prediction feature Z:

$$Z = a_G \cdot Z_G + a_C \cdot Z_C + a_Q \cdot Z_Q. \tag{8}$$

4.3. Feature graph selection

To minimize bias and infer a reliable graph structure when estimating graph structure, multifaceted information must be compiled. After performing the l-th neighborhood aggregation iteration, AMBGAT provides local-to-global information about the nodes by allowing us to capture the structural information within the lth-order neighborhood of the nodes.

In particular, an observation set $O = \left\{O^{(0)}, O^{(1)}, \ldots, O^{(l)}\right\}$ is constructed based on the node feature matrix $X = \left\{H^{(0)}, H^{(1)}, \ldots, H^{(l)}\right\}$ generated by the l neighborhood aggregation of the AGBGN model, where $O^{(h)} \in \mathbb{R}^{n\times n}$ is the adjacency matrix of the kNN graph generated by the convolutional feature $H^{(h)}$ at the h-th layer and describes the neighborhood similarity at the h-th layer. For features x_i and x_j corresponding to nodes i and j in layer h, the expression for the similarity matrix is:

$$\mu_{ij} = \frac{x_i \cdot x_j}{|x_i|\left|x_j\right|}, \tag{9}$$

where calculating the cosine similarity of two vectors x_i and x_j yields the similarity matrix μ_{ij}, the first k similar nodes are then selected evenly for each node to build the concatenated edges, yielding a kNN graph in each layer.

These observation graphs depict the optimal graph structure from various vantage points, and the original neighbor matrix A is also an observation, therefore the observation set $O = \left\{A, O^{(0)}, O^{(1)}, \ldots, O^{(l)}\right\}$. The observation set O, the predicted values Z, and the labels Y_L are all put into the Bayesian estimator to infer a good idea of the posterior distribution of the estimated graphs. Explain in the subsection that follows how the estimated graph was generated.

4.4. Graph Estimator

Until now, the generation of the observation set O and the most pertinent predictions Z for the data labels have been described. However, what is the estimated graph Q on the data set? Can the observation graphs in the results of an observation be directly treated as optimal estimated graphs? Although the observations describe the optimal graph structure from a variety of perspectives, they may be unreliable or insufficient, and their accuracy cannot be predicted. Therefore, it is assumed that an optimal symmetric adjacency matrix with GNN properties has been generated, and the probability of mapping these observation sets onto the symmetric adjacency matrix is computed by Bayesian inference, so that a computational inversion is achieved by computing the posterior distribution of the graph structure, allowing us to achieve our goal.

In recent related literature [38, 39], the use of stochastic block model (SBM) to generate symmetric adjacency matrices with stronger community structure [59] has been mentioned, where the homogeneity of the generated graph structure can be restricted by fitting the intra- and inter-community parameters in the block model. The standard SBM is used, which is typically effective for generating estimated graphs and represents the estimated symmetric adjacency matrix as the estimated graph S.

Specifically, the generation of the estimated graph S employs a probability distribution $P(S \mid \Omega, Z, Y_L)$. where Ω is a parameter of the SBM that represents the probability of edges connecting nodes within and between communities. The community of nodes dictates the probability of connecting edges between distinct nodes. For instance, the probability of generating an edge between nodes v_i and v_j belonging to communities c_i and c_j is Ω_{ij}. Given the parameters Ω, prediction Z, and label Y_L, it is possible to formalize the computation of the estimated graph S as follows:

$$P(S \mid \Omega, Z, Y_L) = \prod_{i<j} \Omega_{c_i c_j}^{S_{ij}} \left(1 - \Omega_{c_i c_j}\right)^{1-S_{ij}}, \quad (10)$$

where

$$c_i = \begin{cases} y_i & \text{if } v_i \in V_L, \\ Z_i & \text{otherwise.} \end{cases} \quad (11)$$

In order to gain more accurate community identification, the real labels are used to substitute the community categories of nodes in the training set when calculating the node communities c. Consequently, the estimated graph S is constructed by calculating the probability Ω_{ij} of linked edges between nodes v_i and v_j.

Although the underlying structure of the estimated graph S is consistent with the nature of the GNN, it is unknown in what form the ideal graph structure exists. Therefore, it is necessary to infer the graph structure by combining as much external observation information as possible. The following is an explanation of how the estimated graph S relates to the set of observations O.

Possible observations are parameterized by two variables: the true positive rate δ and the true negative rate φ. Where δ represents the probability of detecting the genuine presence of an edge in an estimated graph S and φ represents the probability of observing the absence of an edge in an estimated graph S. The probability mapped to each observation is expressed through $P(O \mid S, \delta, \varphi)$, the condition expresses the presence or absence of a contiguous edge in the estimated graph and assumes that the observations of the edge are independent Bernoulli identically distributed random variables, an assumption that proves to be feasible [60]. We assume a total of M (i.e., $|O|$) observations and, on the basis of these observations, determine the presence of an edge E_{ij} times and no edge $M - E_{ij}$ times. On the basis of the preceding hypotheses, the probability form is derived:

$$P(O \mid S, \delta, \varphi) = \prod_{i<j} \left[\delta^{E_{ij}}(1-\delta)^{M-E_{ij}}\right]^{S_{ij}} \times \left[\varphi^{E_{ij}}(1-\varphi)^{M-E_{ij}}\right]^{1-S_{ij}}, \quad (12)$$

if the estimated graph S contains an edge, then $S_{ij} = 1$.

The procedure for obtaining the estimated graph S and mapping it to the observation set O is outlined. However, it is challenging to directly calculate the posterior probability $P(S, \Omega, \delta, \varphi \mid O, Z, Y_L)$ of the estimated graph S. In the following part, the use of Bayesian methods to determine the posterior probability of the estimated graph S is described. The Bayesian inference is expressed as follows:

$$P(S, \Omega, \delta, \varphi \mid O, Z, Y_L) = \frac{P(O \mid S, \delta, \varphi) P(S \mid \Omega, Z, Y_L) P(\Omega) P(\delta) P(\varphi)}{P(O, Z, Y_L)}, \quad (13)$$

wherein the probability of each parameter is assumed to be independent.

By summing all possible values of the estimated graph S, equations for the posterior probabilities of the parameters Ω, δ, and φ can be obtained:

$$P(\Omega, \delta, \varphi \mid O, Z, Y_L) = \sum_S P(S, \Omega, \delta, \varphi \mid O, Z, Y_L). \quad (14)$$

Maximizing the posterior probabilities of the parameters Ω, δ, and φ yields the maximum a posteriori (MAP) estimations of these parameters. On this basis, the symmetric adjacency matrix Q of the estimated graph S can be computed.

$$Q_{ij} = \sum_S q(S) S_{ij}, \quad (15)$$

where the symmetric adjacency matrix Q_{ij} represents the posterior probability of the existence of an edge between nodes i and j and the confidence level of this edge.

4.5. Iterative optimization

The convolution module parameters ϑ and the predicted symmetric adjacency matrix Q are interdependent. Consequently, iterative optimization is challenging. The aforementioned issues are overcome by employing an alternate optimization of the parameters ϑ and Q.

4.5.1. Update ϑ

The two output features Z_{CG} and Z_{CQ} are further constrained to increase their commonality in order to capture their shared information, and the similarity matrix of the aforesaid two output features is expressed as:

$$L_G = \|Z_{CG}\|_2 \cdot \|Z_{CG}\|_2^T, \tag{16}$$

$$L_Q = \left\|Z_{CQ}\right\|_2 \cdot \left\|Z_{CQ}\right\|_2^T. \tag{17}$$

To highlight their commonalities, the following constraints are generated:

$$\mathcal{L}_c = \left\|L_G - L_Q\right\|_F^2. \tag{18}$$

Since the two output features Z_G and Z_{CG} are learned from the same graph (the original topological graph G), the Hilbert-Schmidt Independence Criterion (HSIC) [61] is applied, a compact and efficient independence measure theory, to ensure that their different information is captured. In machine learning-related literature, the usefulness of the idea has been demonstrated [62, 63]. Constraining the aforementioned two output features by HSIC is stated as:

$$HSIC\left(Z_G, Z_{CG}\right) = (n-1)^{-2}\,\mathrm{tr}\left(RK_GRK_{CG}\right), \tag{19}$$

where K_G and K_{CG} are Gram matrices with $k_{G,ij} = k_G\left(z_G^i, z_G^j\right)$ and $k_{CG,ij} = k_{CG}\left(z_{CG}^i, z_{CG}^j\right)$ characteristics, respectively, and $R = I - \left[\left(ee^G\right)/n\right]$, where I is a unit matrix and e is a column vector consisting of a single element. The inner product kernel functions of K_G and K_{CG} are employed in the implementation.

The two output features Z_Q and Z_{CQ} are learned in the same graph (estimation graph Q), and their $HSIC\left(Z_Q, Z_{CQ}\right)$ is identical to the approach applied to the output features Z_G and Z_{CG}.

To capture personality information, the following constraints are generated:

$$\mathcal{L}_p = HSIC\left(Z_G, Z_{CG}\right) + HSIC\left(Z_Q, Z_{CQ}\right). \tag{20}$$

Next, the predicted features Z generated from Equation 8 are applied to the linear transform and $softmax$ function in order to express the predicted $\hat{Z}$ of the training node V_L as follows:

$$\hat{Z} = softmax(W \cdot Z + b). \tag{21}$$

We concentrated on the semi-supervised node classification problem, evaluating the cross-entropy error of prediction $\hat{Z}$ on all training nodes V_L:

$$\mathcal{L}_l = -\sum_{v_i \in V_L} Y_i \ln \hat{Z}_i. \tag{22}$$

Finally, the model's global objective function can be obtained using the following equation:

$$\mathcal{L}_\vartheta = \mathcal{L}_l + \mu\mathcal{L}_c + \tau\mathcal{L}_p, \tag{23}$$

where μ and τ represent the parameters for the two constraint terms $\mathcal{L}_c$ and $\mathcal{L}_p$. For the overall objective function $\mathcal{L}_\vartheta$, stochastic gradient descent can be used to learn the model's parameters ϑ.

4.5.2. Update Q

In recent studies, some researchers [64] have utilized the expectation maximization (EM) algorithm [65, 66] to efficiently train the joint distribution of object labels in the field of GNNs, achieving satisfactory results. In order to optimize the symmetric adjacency matrix Q, the EM algorithm and maximized Equation 14 are also employed.

E-step (a.k.a., inference procedure). Direct maximization of Equation 14 is typically difficult to solve. Hence, Equation 14 is solved using Jensen's inequality as follows:

$$\log P\left(\Omega, \delta, \varphi \mid O, Z, Y_L\right) \geq \sum_S q(S) \log \frac{P\left(S, \Omega, \delta, \varphi \mid O, Z, Y_L\right)}{q(S)}, \tag{24}$$

where $q(S)$ is any nonnegative function of the estimated graph S fulfilling $\sum_S q(S) = 1$ and is a probability distribution on the estimated graph S.

Maximizing the right side of equation 24 necessitates that the equation hold true and can be derived as follows:

$$q(S) = \frac{P\left(S, \Omega, \delta, \varphi \mid O, Z, Y_L\right)}{\sum_S P\left(S, \Omega, \delta, \varphi \mid O, Z, Y_L\right)}. \tag{25}$$

By combining Equation 10 and Equation 12 into Equation 25 and eliminating the constants from the fraction, the following expression for $q(S)$ is obtained:

$$\begin{aligned} q(S) &= \frac{\prod_{i<j}\left[\Omega_{c_ic_j}\delta^{E_{ij}}(1-\delta)^{M-E_{ij}}\right]^{S_{ij}}\left[\left(1-\Omega_{c_ic_j}\right)\varphi^{E_{ij}}(1-\varphi)^{M-E_{ij}}\right]^{1-S_{ij}}}{\sum_S \prod_{i<j}\left[\Omega_{c_ic_j}\delta^{E_{ij}}(1-\delta)^{M-E_{ij}}\right]^{S_{ij}}\left[\left(1-\Omega_{c_ic_j}\right)\varphi^{E_{ij}}(1-\varphi)^{M-E_{ij}}\right]^{1-S_{ij}}} \\ &= \prod_{i<j}\frac{\left[\Omega_{c_ic_j}\delta^{E_{ij}}(1-\delta)^{M-E_{ij}}\right]^{S_{ij}}\left[\left(1-\Omega_{c_ic_j}\right)\varphi^{E_{ij}}(1-\varphi)^{M-E_{ij}}\right]^{1-S_{ij}}}{\Omega_{c_ic_j}\delta^{E_{ij}}(1-\delta)^{M-E_{ij}} + \left(1-\Omega_{c_ic_j}\right)\varphi^{E_{ij}}(1-\varphi)^{M-E_{ij}}}. \end{aligned} \tag{26}$$

In the E-step, fix the parameters Ω, δ, φ to maximize $q(S)$. Then, further maximizing the right-hand side of Equation 24 will provide us with an estimate of MAP.

M-step (a.k.a., learning procedure). Differentiation yields the maximum of the parameters. Taking the derivatives of the right-hand side of Equation 24 while holding $q(S)$ constant and assuming uniform prior yields, the following is obtained:

$$\sum_S q(S) \sum_{i<j}\left(\frac{S_{ij}}{\Omega_{c_ic_j}} - \frac{1-S_{ij}}{1-\Omega_{c_ic_j}}\right) = 0, \tag{27}$$

$$\sum_S q(S) \sum_{i<j} S_{ij}\left(\frac{E_{ij}}{\delta} - \frac{M-E_{ij}}{1-\delta}\right) = 0, \tag{28}$$

$$\sum_S q(S) \sum_{i<j}\left(1-S_{ij}\right)\left(\frac{E_{ij}}{\varphi} - \frac{M-E_{ij}}{1-\varphi}\right) = 0, \tag{29}$$

where Equations 27, 28 and 29 represent the greatest a posteriori estimates of Ω, δ, and φ, respectively.

Specifically, with respect to the specific calculation, the probability Ω_{ab} of the existence of an edge for communities a and b is calculated by averaging the

probabilities of each edge between all nodes in these two communities, denoted as:

$$\Omega_{ab} = \begin{cases} \frac{\psi_{ab}}{\phi_a \phi_b} & \text{if } a \neq b, \\ \frac{2\psi_{aa}}{\phi_a(\phi_a - 1)} & \text{otherwise,} \end{cases} \tag{30}$$

where ϕ_a denotes the number of nodes in community a, and ψ_{ab} is the sum of the probability of the existence of edges between nodes in communities a and b on the neighbor matrix Q. For the specific calculation of δ and φ, denoted as:

$$\delta = \frac{\sum_{i<j} Q_{ij} E_{ij}}{M \sum_{i<j} Q_{ij}}, \tag{31}$$

$$\varphi = \frac{\sum_{i<j} \left(1 - Q_{ij}\right) E_{ij}}{M \sum_{i<j} \left(1 - Q_{ij}\right)}. \tag{32}$$

To determine the value of the symmetric adjacency matrix Q, substitute the equation 26 derived in E-step into equation 15, denoted as follows:

$$Q_{ij} = \frac{\Omega_{c_i c_j} \delta^{E_{ij}} (1-\delta)^{M - E_{ij}}}{\Omega_{c_i c_j} \delta^{E_{ij}} (1-\delta)^{M - E_{ij}} + \left(1 - \Omega_{c_i c_j}\right) \varphi^{E_{ij}} (1-\varphi)^{M - E_{ij}}}. \tag{33}$$

Calculating the posterior distribution $q(S)$ can be simplified by determining the value of the symmetric adjacency matrix Q_{ij}:

$$q(S) = \prod_{i<j} Q_{ij}^{s_{ij}} \left(1 - Q_{ij}\right)^{1 - s_{ij}}. \tag{34}$$

In other words, the probability distribution on the estimated graph is the product of the independent Bernoulli distributions of the individual edges, where the Bernoulli parameter Q_{ij} represents the uncertainty in both the structure of the graph and the structure itself.

In M-step, the parameters Ω, δ, φ are learned with $q(S)$ held constant. Consequently, it is natural to employ an EM algorithm to complete the alternating calculation of parameter values Ω, δ, φ and the posterior distribution $q(S)$ on the graph structure, and to repeat the iterative process until convergence.

Post-processing. The estimated symmetric adjacency matrix Q expresses the likelihood of edges linking all nodes. To increase the quality of the generated nodes' edges, the sparse symmetric adjacency matrix Q^s is created by setting a threshold γ to mask off smaller-than-threshold elements from Q. This matrix is denoted as:

$$Q_{ij}^S = \begin{cases} Q_{ij} & \text{if } Q_{ij} > \gamma, \\ 0 & \text{otherwise .} \end{cases} \tag{35}$$

Algorithm 1 AMBGAT

Input: original topology graph G_O, feature graph G_F, feature matrix A, labels Y_L, threshold λ, γ, iterations $\{\rho_I, \rho_L\}$, parameter μ, τ, k.

Output: estimated graph Q^s, model parameters ϑ.

```
Initialization: initialize parameter δ, φ, ϑ, Ω and update ϑ after initializing observation set O with G and K;
for i = 1 to ρ_I do
    while |δ − δ^old| > λ or |φ − φ^old| > λ do
        % Graph Estimator Training
        Ω^old = Ω, δ^old = δ, φ^old = φ;
        Calculate Ω, δ and φ by eq.(30), eq.(31), eq.(32), respectively;
        Update Q by eq.(33);
    end while
    Using the threshold γ to obtain Q^s by eq.(35);
    Replace the estimated graph by Q^s;
    for l = 1 to ρ_L do
        % Classifiers Training
        Calculating predictions by eq.(21);
        Update ϑ by eq.(23);
    end for
end for
return Q^s and ϑ.
```

4.5.3. Algorithm description

Based on the preceding criteria for alternate iterative optimization of parameters ϑ and Q, Algorithm 1 describes the resulting algorithm. Specifically, the model first enters the preprocessing phase, which initializes all parameters at random before generating the observation set O. For the first iteration optimization, the estimation graph Q is not used, as the first iteration cannot provide additional information that would cause bias in the estimation graph Q. As the initial input, a reliable feature graph (computed from the features of the original nodes) and the original topology graph are employed to initialize the observation set O. Next, the symmetric adjacency matrix Q^s is estimated from the observation set O, which in turn helps the optimization of the model parameters ϑ. More precise model parameters ϑ can produce more accurate observations for graph estimation Q^s. Finally, until convergence is reached, this alternating iterative strategy optimizes the model parameters ϑ and estimated graph Q^s positively.

5. Experiments

Extensive tests are conducted in this section to determine the efficacy of AMBGAT for the account classification task in an Ethernet transaction network. First, the experimental setting is discussed, including the dataset, baseline, and implementation particulars. Then, AMBGAT is compared to previous approaches. Next, multiple variants of AMBGAT are compared

Table 5
Node classification results. (index: ACC and Macro-F1; bold: best)

Algorithm	10 labels/class		20 labels/class		40 labels/class	
	ACC	Macro-F1	ACC	Macro-F1	ACC	Macro-F1
Deepwalk	0.271	0.267	0.392	0.373	0.434	0.428
Chebyshev	0.265	0.263	0.427	0.402	0.438	0.422
GCN	0.397	0.374	0.435	0.413	0.478	0.451
GAT	0.417	0.406	0.502	0.468	0.581	0.505
MixHop	0.405	0.392	0.519	0.489	0.662	0.641
H2GCN	0.609	0.602	0.659	0.645	0.676	0.653
AM-GCN	0.709	0.703	0.748	0.747	0.796	0.769
GEN	0.715	0.718	0.734	0.726	0.835	0.813
CoGSL	0.742	0.726	0.805	0.769	0.842	**0.837**
*k*NN-GAT	0.545	0.537	0.584	0.561	0.628	0.586
AMBGAT	**0.794**	**0.73**	**0.818**	**0.776**	**0.853**	0.836

Table 6
Node classification results. (index: Recall and Micro-F1; bold: best)

Algorithm	10 labels/class		20 labels/class		40 labels/class	
	Recall	Micro-F1	Recall	Micro-F1	Recall	Micro-F1
Deepwalk	0.315	0.279	0.446	0.398	0.457	0.493
Chebyshev	0.318	0.268	0.48	0.407	0.428	0.491
GCN	0.403	0.406	0.482	0.427	0.454	0.48
GAT	0.449	0.438	0.526	0.511	0.567	0.542
MixHop	0.445	0.405	0.524	0.52	0.702	0.719
H2GCN	0.634	0.655	0.712	0.66	0.706	0.716
AM-GCN	0.74	0.737	0.761	0.757	0.756	0.866
GEN	0.778	0.743	0.719	0.75	0.811	0.864
CoGSL	0.714	0.748	0.808	0.793	0.821	0.867
*k*NN-GAT	0.605	0.541	0.582	0.612	0.642	0.667
AMBGAT	**0.788**	**0.754**	**0.851**	**0.797**	**0.857**	**0.871**

and the effectiveness of the adaptive mechanism is discussed, followed by an examination of the effects under node attack. Finally, the hyperparameter experiments of the model are deeply investigated.

5.1. Experimental setup

5.1.1. Dataset

Ethereum is now the leading platform for smart contracts on the blockchain. Due to the anonymity of Ethereum, the names of all users are concealed under anonymous accounts, fostering illicit and criminal behavior. By embedding actual ethereum transaction data into the ethereum transaction network, the classification of ethereum accounts may be converted into the classification of nodes in the ethereum transaction network. Classification of nodes in the Ethernet transaction network can help us identify various account kinds, hence enhancing the platform's transparency and auditability. The details of the Ethernet dataset used to test the performance of our proposed AMBGAT are detailed in Table 2. The experiments focus on typical semi-supervised learning while dividing multiple training sets, including specifying 10 labeled nodes, specifying 20 labeled nodes, and 40 labeled nodes for each category of the training set, and always specifying 1000 labeled nodes for the test set.

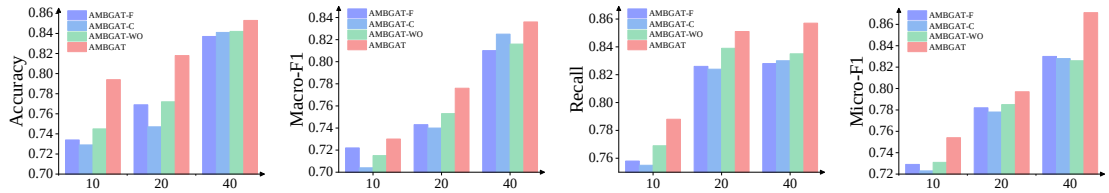
5.1.2. Baselines

AMBGAT is compared with three exemplary GNNs, including a network embedding algorithm, six graph neural network-based algorithms, and two graph structure learning algorithms.

- **DeepWalk** [53] is a traditional algorithm for network embedding. It uses random walking to gather context information and the co-occurrence relationship between graph nodes to learn the vector representation of nodes.
- **Chebyshev** [26] is a graph convolution network employing the Chebyshev filter.
- **GCN** [23] is a semi-supervised graph embedding method that learns the representation of nodes by aggregating their neighbors.
- **GAT** [35] employs an attention mechanism to assign different importance to different neighbors of a node in the process of node feature aggregation.
- **MixHop** [67] is a high-order neighbor-focused graph convolution neural network.
- **H2GCN** [57] is a graph neural network with superior performance in networks with low homogeneity. Self-embedding and neighborhood-embedding separation, with an emphasis on high-order neighbors, comprise the key design.
- **AM-GCN** [62] is a graph convolution network based on multi-views that learns node embedding from node features and topology.
- **GEN** [38] is a GNN based on learning graph structures. It adapts to the GNN mechanism by constructing estimation graphs and estimating more precise graph structures by employing a wide range of information.
- **CoGSL** [41] is a GNN based on graph structure learning, which learns the minimally sufficient graph structure to enhance GNN performance.
- ***k*NN-GAT** utilizes the *k*NN algorithm to build the *k*-nearest neighbor graph of the dataset as the adjacency matrix of the GAT to provide a more exhaustive comparison.

5.1.3. Implementation details

The model's implementation consists of two GAT layers with the same three hidden layer dimensions and output layer dimensions. Each layer contains $K = 4$ attention-head computations. The model employs a learning rate of 0.01, a weight attenuation of $5e-4$, a dropout rate of 50% per layer, and the Adam optimizer for training. In addition, select the embedding dimension of the hidden layer from {512, 768} and the embedding dimension of the output layer from {16, 32, 128, 256} for the hyperparametric option. The feature graph $K \in \{2, ..., 10\}$, threshold $\lambda \in \{0.1, 0.01, 0.001, 0.0001\}$, and threshold



(a) Accuracy (b) Macro-F1 (c) Recall (d) Micro-F1

Fig. 4. The results of AMGBAT and its three variants on the Ethereum transaction dataset.

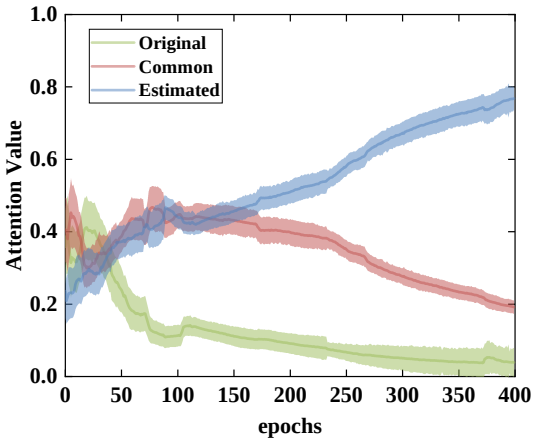


Fig. 5. The changing trend of attention during epochs.

$\gamma \in \{0.1 \ldots 0.9\}$. Coefficient of consistency constraint $\mu = \{0.1, 0.001, 0.0001\}$ and coefficient of independence $\tau = \{1e-10, 5e-9, 1e-9, 5e-8, 1e-8\}$. During the model training and testing, only the classification performance of 3785 labeled nodes is considered. The number of model optimization iterations is set to 200, and the parameters with the highest verification accuracy are stored for testing. Five separate trials with different random seeds were done for each technique, and the average Accuracy (ACC), Macro-F1 score, Recall and Micro-F1 score were reported to evaluate the model's performance.

5.2. *Node classification*

The performance of AMBGAT was evaluated for a semi-supervised node classification task on the Ethernet trading network, and the results are shown in Tables 5 and 6. The scenario of randomly assigning 10, 20, and 40 genuine labels to each class of the dataset is examined. With 10 and 20 labels assigned to each class, the training set is comprised of the top 10 and 20 labels in the case of 40 partitions, while the test set remains unchanged.

Based on the configuration described above, the following observations can be made:

- The proposed AMBGAT outperforms other baseline methods for various true label rates, indicating that the graph estimation model can improve the aggregation capability of GNNs by learning topologies that correspond to the ground truth, thereby enhancing the robustness of node classification performance.
- The enormous performance advantage of AMBGAT over traditional GAT suggests a synergistic optimization and mutual reinforcement between the graph structure estimator and feature learning.
- Our performance is more noticeable compared to kNN-GAT, indicating that Bayesian is superior to k-NN in predicting the ground-truth-compliant graph structure, hence demonstrating the need for Bayesian inference to improve the classification capacity of nodes.
- Compared to existing graph structure learning frameworks, our performance improvement demonstrates that AMBGAT is effective on Ethereum trade networks with low homogeneity and is able to learn the most pertinent node feature information for the node classification task.

5.3. *Ablation analysis*

Examine AMBGAT and its three versions against the Ethereum transaction dataset to validate the model's plausibility and validity.

- AMBGAT-F: Extract feature information from the estimated graph using only the personality convolution module.
- AMBGAT-C: Extract common features from the original and estimated graphs using only the common convolution module.
- AMBGAT-WO: The individual personality convolution modules are employed to extract the features from the respective graphs, rather than the common convolution module.

Figure 4 demonstrates: (1) AMBGAT outperforms the other three examples on the Ethereum transaction dataset, demonstrating the effectiveness of learning both the personality and common convolution information on the estimated graph. (2) AMBGAT outperforms AMBGAT-C, indicating that meaningful feature information can be learned in both graph structures independently. (3) The performance of AMBGAT-F is

not ideal, indicating that the original graph structure still contains useful information.

5.4. Analysis of adaptive mechanism

To determine the efficacy of the adaptive mechanism described in subsection 4.2, which is used to extract the information most pertinent to the node labels, the attention distribution trends of the original graph, the estimated graph, and their common information are analyzed.

Figure 5 depicts the analysis of the mean and standard deviation change process of attention values during model training at 40 label rates on the Ethernet transaction dataset. The X-axis represents the change in epoch during the first iteration of model parameter optimization, while the Y-axis represents the weight of the attention values. As the training epoch advances, the attention values of the original graph, the estimated graph, and their combination diverge. The attention value of the estimated graph continues to increase as training progresses, while the attention value of the real graph continues to decrease. This is compatible with the experimental view of evaluating the Ethernet transaction network, in which the original Ethernet transaction network is less homogeneous and the estimated graph structure is more homogeneous and better suited to the classification task. In summary, this section's tests demonstrate that the proposed model may adaptively assign a larger attention value to the embedding on the estimated graph and the embedding on the original graph based on their relative relevance.

5.5. Robustness analysis

In this subsection, the defensive performance of several models against the Ethereum transaction dataset is assessed. Specifically, according to the approach of [68], the edges of the Ethereum transaction dataset are attacked, and random additions and deletions are made to the edges. GNNs that support structure learning are typically more robust than other GNNs because they are able to learn graph structures that are actually true. The poisoning attack [69] is selected. First, an attacked Ethereum trading network is constructed, and then it is utilized to train the model. GAT is chosen as the representative of the conventional GNN model, and GEN and CoGSL as the representatives of the graph structure learning model.

For the addition of edges, the original Ethereum transaction dataset is randomly added with 25%, 50%, and 75% of the original number of edges (if there is no such edge). For the deletion of edges, 5%, 10%, and 15% of the edges in the original Ethereum transaction dataset are randomly deleted without adding additional isolated nodes. In addition, since both CoGSL and the proposed AMBGAT need two inputs at the beginning of training, in order to ensure accurate evaluation, both inputs are attacked by the same percentage. All experiments were trained five times, and the average accuracy was reported, as shown in figures 6 and 7. Furthermore, AMBGAT-o and AMBGAT-f indicate that the input of the original graph and the input of the feature graph are attacked, respectively, while AMBGAT indicates that both inputs are attacked.

Compared with other models, the three cases of AMBGAT achieved better results in both scenarios. GAT performance is poor in the scene where edges are added. It is speculated that simple graph neural networks may not be able to better deal with misleading additional random noise. It is also found that with the increase of disturbance, the performance gap becomes more obvious, which indicates that AMBGAT is more effective in identifying accounts with camouflage ability in the Ethereum transaction network.

5.6. Sensitivity of hyper-parameters

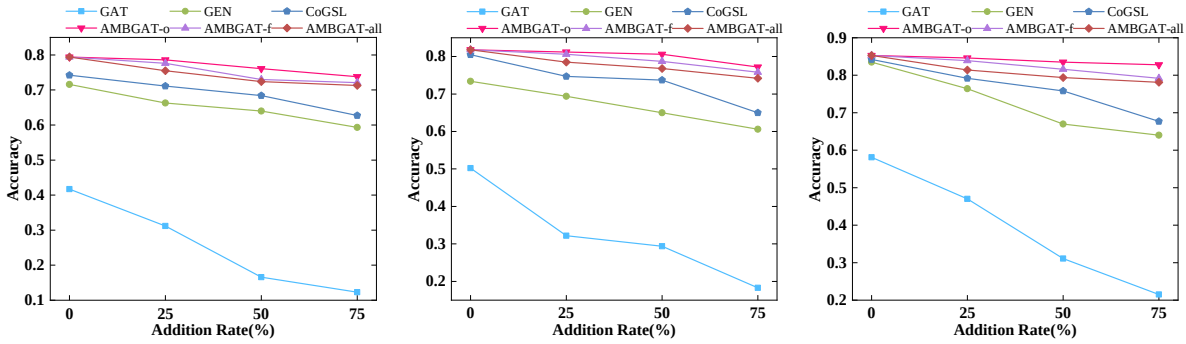
In this subsection, the effects of hyperparameter changes on the performance of the model on the Ethereum transaction dataset are investigated, including the coefficient k in the feature graph, the threshold λ in the EM algorithm, the threshold γ in Equation 35, the consistency coefficient μ, and the independence coefficient τ in Equation 23, for label rates of 10, 20, and 40, respectively. Adjusting k between 2 and 10, the threshold λ between $1e-5$ and 0.1, the threshold γ between 0.1 and 0.9, the coefficient μ between 0 and 10000, and the coefficient τ between 0 and $1e-4$. The experimental outcomes are depicted in Figures 8 and 9.

Coefficient k within a feature graph. The results indicate that when k goes from 2 to 10, the model performance increases and subsequently decreases. This may be because a small value of k leads to information loss, whereas a large value of k introduces noisier edges.

Threshold λ is utilized by the EM algorithm. The threshold value λ has an impact on the convergence rate of the employed EM algorithm. The optimal performance of the model is reached when the threshold λ is approximately 0.01; any other value influences model performance.

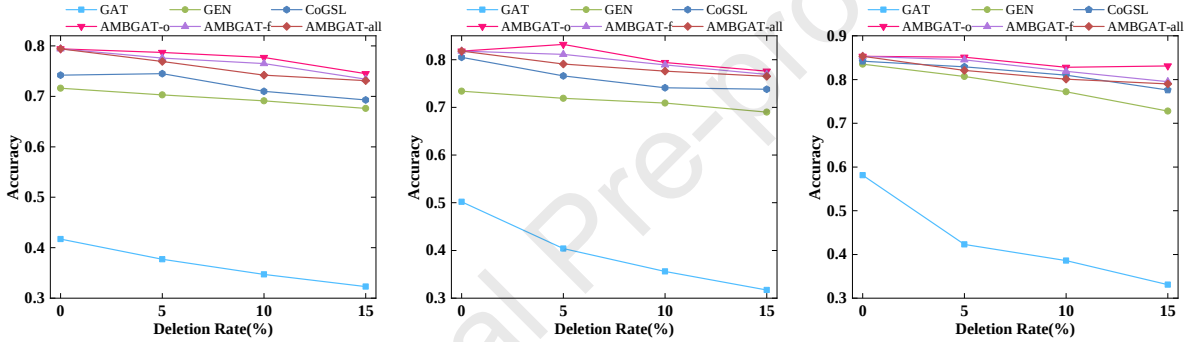
Threshold γ. The effect of the threshold γ is verified in Equation 35. If the threshold γ is set too high, the estimated graph structure disregards more meaningful edges, but if it is set too low, the estimated graph structure includes more noisy edges. The optimal value of the curve with threshold γ may be observed on the left side of the X-axis. This is typically owing to the low homogeneity of the dataset, which results in greater variations between observations and, thus, lower edge confidence.

Consistency coefficient μ. The effect of the coefficient μ is verified in Equation 23, and it is observed that as the coefficient μ increases, the model performance improves and then declines. The performance of the model is ideal for the coefficient μ in the range



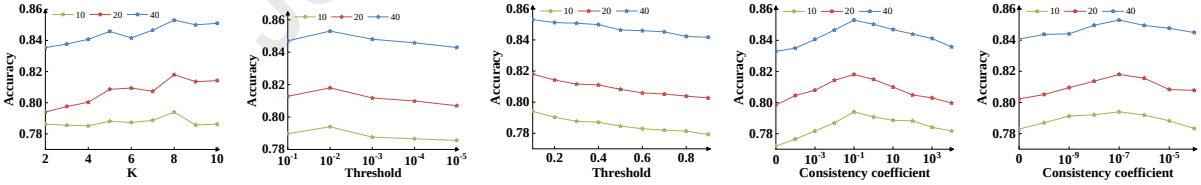
(a) 10 labels (b) 20 labels (c) 40 labels

Fig. 6. Results from various models in scenarios where random edges are added.



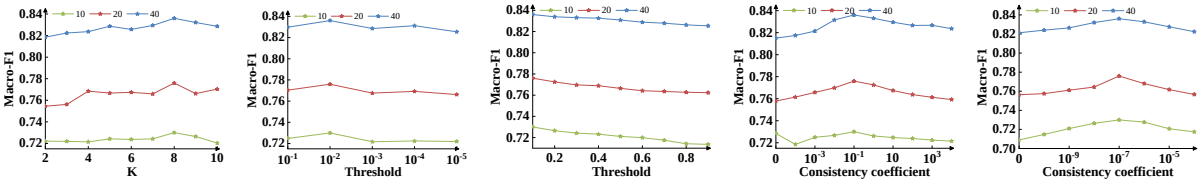
(a) 10 labels (b) 20 labels (c) 40 labels

Fig. 7. Results from various models in scenarios where random edges are deleted.



(a) k in kNN (b) Threshold π (c) Threshold γ (d) Coefficient μ (e) Coefficient τ

Fig. 8. Hyper-parametric analysis of the Accuracy index.



(a) k in kNN (b) Threshold π (c) Threshold γ (d) Coefficient μ (e) Coefficient τ

Fig. 9. Hyper-parameter analysis of the Macro-F1 index.

of $1e-4$ to $1e+4$, while other values degrade its performance.

Independence coefficient τ. The effect of coefficient τ is also verified in Equation 23, and experimental data indicates that the variation in model performance is rather constant as coefficient τ varies.

6. Conclusion

In this work, a graph neural network is used to make the Ethereum transaction platform safer and a better place for Internet of Things devices to store and share data. Specifically, an AMBGAT model based on graph structure learning and multi-view ideas is proposed to identify different types of accounts in Ethereum. By analyzing the Ethereum transaction network, we find that accounts with transaction relationships differ significantly from each other in terms of both label and feature information. The analysis results motivate the need to design a new graph structure to efficiently learn node embedding information. Specifically, we base our work on a node-level attentional convolution module to learn to enhance features for nodes; estimate the new graph structure based on Bayesian inference while extracting specific and common embedding information from the estimated graph, the original topological graph, and their combinations; use the attentional mechanism to learn the importance weights of these three node embeddings; and finally fuse the node feature information that is most beneficial to the classification task. Numerous experiments confirm the effectiveness of the proposed AMBGAT in the classification of Ethernet accounts and confirm that the model possesses the ability to fuse out the node features that are most beneficial for downstream tasks. The Ethereum platform is intrinsically trustworthy and tamper-proof. If platform fraud is eliminated, the Ethereum trading platform will be recognized as a reliable platform for the storage and transaction of Internet of Things data.

A future work should investigate the security of IoT data interaction in greater depth and expand AMBGN to the dynamic Ethereum transaction network containing timing information. However, it remains problematic to learn the parameters of AMBGAT in a time-series manner, so these considerations inspire a more sophisticated training strategy for adaptation.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (62072391, 62066013), Cooperation Project between School and Locality of Yantai. The corresponding author is Zhaowei Liu.

References

[1] L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, Computer networks 54 (15) (2010) 2787–2805.

[2] K. S. Mohamed, Iot cloud computing, storage, and data analytics, in: The Era of Internet of Things, Springer, 2019, pp. 71–91.

[3] Y. Wu, Y. Lyu, Y. Shi, Cloud storage security assessment through equilibrium analysis, Tsinghua Science and Technology 24 (6) (2019) 738–749.

[4] M. Iansiti, K. R. Lakhani, The truth about blockchain, Harvardbusiness review 95 (01) (2017) 118–127.

[5] C. Esposito, A. De Santis, G. Tortora, H. Chang, K.-K. R. Choo, Blockchain: A panacea for healthcare cloud-based data security and privacy?, IEEE Cloud Computing 5 (1) (2018) 31–37.

[6] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, L. Njilla, Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability, in: 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, IEEE, 2017, pp. 468–477.

[7] H. Shafagh, L. Burkhalter, A. Hithnawi, S. Duquennoy, Towards blockchain-based auditable storage and sharing of iot data, in: Proceedings of the 2017 on cloud computing security workshop, 2017, pp. 45–50.

[8] G. Wood, et al., Ethereum: A secure decentralised generalised transaction ledger, Ethereum project yellow paper 151 (2014) (2014) 1–32.

[9] M. Yutaka, Y. Zhang, M. Sasabe, S. Kasahara, Using ethereum blockchain for distributed attribute-based access control in the internet of things, in: 2019 IEEE Global Communications Conference, IEEE, 2019, pp. 1–6.

[10] A. Raj, K. Maji, S. D. Shetty, Ethereum for internet of things security, Multimedia Tools and Applications 80 (12) (2021) 18901–18915.

[11] S. Mehedi, A. A. M. Shamim, M. B. A. Miah, Blockchain-based security management of iot infrastructure with ethereum transactions, Iran Journal of Computer Science 2 (3) (2019) 189–195.

[12] H. Sun, S. Hua, E. Zhou, B. Pi, J. Sun, K. Yamashita, Using ethereum blockchain in internet of things: A solution for electric vehicle battery refueling, in: International Conference on Blockchain, Springer, 2018, pp. 3–17.

[13] S. Lee, C. Yoon, H. Kang, Y. Kim, Y. Kim, D. Han, S. Son, S. Shin, Cybercriminal minds: an investigative study of cryptocurrency abuses in the dark web, in: 26TH ANNUAL NETWORK AND DISTRIBUTED SYSTEM SECURITY SYMPOSIUM, Internet Society, 2019, pp. 1–15.

[14] L. Chen, J. Peng, Y. Liu, J. Li, F. Xie, Z. Zheng, Phishing scams detection in ethereum transaction network, ACM Transactions on Internet Technology 21 (1) (2020) 1–16.

[15] C. F. Torres, M. Steichen, et al., The art of the scam: Demystifying honeypots in ethereum smart contracts, in: 28th USENIX Security Symposium (USENIX Security 19), 2019, pp. 1591–1607.

[16] J. Wu, J. Liu, W. Chen, H. Huang, Z. Zheng, Y. Zhang, Detecting mixing services via mining bitcoin transaction network with hybrid motifs, IEEE Transactions on Systems, Man, and Cybernetics: Systems 52 (4) (2022) 2237–2249. doi:10.1109/TSMC.2021.3049278.

[17] W. Chen, Z. Zheng, J. Cui, E. Ngai, P. Zheng, Y. Zhou, Detecting ponzi schemes on ethereum: Towards healthier blockchain technology, in: Proceedings of the 2018 world wide web conference, 2018, pp. 1409–1418.

[18] H. Chen, M. Pendleton, L. Njilla, S. Xu, A survey on ethereum systems security: Vulnerabilities, attacks, and defenses, ACM Computing Surveys (CSUR) 53 (3) (2020) 1–43.

[19] S. Farrugia, J. Ellul, G. Azzopardi, Detection of illicit accounts over the ethereum blockchain, Expert Systems with Applications 150 (2020) 113318.

[20] T. Hu, X. Liu, T. Chen, X. Zhang, X. Huang, W. Niu, J. Lu, K. Zhou, Y. Liu, Transaction-based classification and detection approach for ethereum smart contract, Information Processing & Management 58 (2) (2021) 102462.

[21] W. Chen, X. Guo, Z. Chen, Z. Zheng, Y. Lu, Phishing scam detection on ethereum: Towards financial security for blockchain ecosystem., in: IJCAI, 2020, pp. 4506–4512.

[22] Y. Wang, Z. Liu, J. Xu, W. Yan, Heterogeneous network representation learning approach for ethereum identity identification, IEEE Transactions on Computational Social Systems.

[23] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: Proceedings of the International Conference on Learning Representations, 2017.

[24] P. W. Holland, K. B. Laskey, S. Leinhardt, Stochastic blockmodels: First steps, Social networks 5 (2) (1983) 109–137.

[25] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and deep locally connected networks on graphs, in: Proceedings of the International Conference on Learning Representations, 2014.

[26] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, Advances in Neural Information Processing Systems 29 (2016) 3844–3852.

[27] L. W. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, Advances in Neural Information Processing Systems 30 (2017) 1024–1034.

[28] J. Chen, T. Ma, C. Xiao, Fastgcn: Fast learning with graph convolutional networks via importance sampling, in: Proceedings of the International Conference on Learning Representations, 2018.

[29] C. Wang, S. Pan, G. Long, X. Zhu, J. Jiang, Mgae: Marginalized graph autoencoder for graph clustering, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 889–898.

[30] T. N. Kipf, M. Welling, Variational graph auto-encoders, *arXivpreprintarXiv* : 1611.07308.

[31] C. Li, M. Welling, J. Zhu, B. Zhang, Graphical generative adversarial networks, Advances in Neural Information Processing Systems 31.

[32] W. Liu, P.-Y. Chen, F. Yu, T. Suzumura, G. Hu, Learning graph topological features via gan, IEEE Access 7 (2019) 21834–21843.

[33] Y. Li, R. Zemel, M. Brockschmidt, D. Tarlow, Gated graph sequence neural networks, in: Proceedings of International Conference on Learning Representations, 2016.

[34] D. Xu, W. Cheng, D. Luo, X. Liu, X. Zhang, Spatio-temporal attentive rnn for node classification in temporal attributed graphs, in: International Joint Conference on Artificial Intelligence, 2019, pp. 3947–3953.

[35] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: Proceedings of the International Conference on Learning Representations, 2018.

[36] A. Salehi, H. Davulcu, Graph attention auto-encoders, in: 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence, 2020, pp. 989–996.

[37] K. Do, T. Tran, T. Nguyen, S. Venkatesh, Attentional multilabel learning over graphs: a message passing approach, Machine Learning 108 (10) (2019) 1757–1781.

[38] R. Wang, S. Mou, X. Wang, W. Xiao, Q. Ju, C. Shi, X. Xie, Graph structure estimation neural networks, in: Proceedings of the Web Conference 2021, 2021, pp. 342–353.

[39] Y. Zhang, S. Pal, M. Coates, D. Ustebay, Bayesian graph convolutional neural networks for semi-supervised classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 5829–5836.

[40] C. Zheng, B. Zong, W. Cheng, D. Song, J. Ni, W. Yu, H. Chen, W. Wang, Robust graph representation learning via neural sparsification, in: International Conference on Machine Learning, 2020, pp. 11458–11468.

[41] N. Liu, X. Wang, L. Wu, Y. Chen, X. Guo, C. Shi, Compact graph structure learning via mutual information compression, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 1601–1610.

[42] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, J. Tang, Graph structure learning for robust graph neural networks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2020, pp. 66–74.

[43] J. You, R. Ying, J. Leskovec, Position-aware graph neural networks, in: International Conference on Machine Learning, 2019, pp. 7134–7143.

[44] S. Zhu, J. Li, H. Peng, S. Wang, L. He, Adversarial directed graph embedding, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, pp. 4741–4748.

[45] R. Li, Z. Liu, Y. Ma, D. Yang, S. Sun, Internet financial fraud detection based on graph learning, IEEE Transactions on Computational Social Systems.

[46] Z. Gu, D. Lin, J. Zheng, J. Wu, C. Hu, Deep learning-based transaction prediction in ethereum, in: International Conference on Blockchain and Trustworthy Systems, Springer, 2021, pp. 30–43.

[47] J. Liu, J. Zheng, J. Wu, Z. Zheng, Fa-gnn: Filter and augment graph neural networks for account classification in ethereum, IEEE Transactions on Network Science and Engineering.

[48] Q. Yuan, B. Huang, J. Zhang, J. Wu, H. Zhang, X. Zhang, Detecting phishing scams on ethereum based on transaction records, in: 2020 IEEE International Symposium on Circuits and Systems, IEEE, 2020, pp. 1–5.

[49] J. Wu, Q. Yuan, D. Lin, W. You, W. Chen, C. Chen, Z. Zheng, Who are the phishers? phishing scam detection on ethereum via network embedding, IEEE Transactions on Systems, Man, and Cybernetics: Systems.

[50] W. Chen, X. Guo, Z. Chen, Z. Zheng, Y. Lu, Phishing scam detection on ethereum: towards financial security for blockchain ecosystem, in: Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, 2021, pp. 4506–4512.

[51] H. Wen, J. Fang, J. Wu, Z. Zheng, Transaction-based hidden strategies against general phishing detection framework on ethereum, in: 2021 IEEE International Symposium on Circuits and Systems, IEEE, 2021, pp. 1–5.

[52] Z. Yuan, Q. Yuan, J. Wu, Phishing detection on ethereum via learning representation of transaction subgraphs, in: International Conference on Blockchain and Trustworthy Systems, Springer, 2020, pp. 178–191.

[53] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 701–710.

[54] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855–864.

[55] Z. Meng, S. Liang, H. Bao, X. Zhang, Co-embedding attributed networks, in: Proceedings of the twelfth ACM international conference on web search and data mining, 2019, pp. 393–401.

[56] T. Huang, Y. Zhang, J. Wu, J. Fang, Z. Zheng, Mg-gcn: Fast and effective learning with mix-grained aggregators for training large graph convolutional networks, ArXiv abs/2011.09900.

[57] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, D. Koutra, Beyond homophily in graph neural networks: Current limitations and effective designs, Advances in Neural Information Processing Systems 33 (2020) 7793–7804.

[58] M. E. J. Newman, Networks: An Introduction, Oxford University Press, 2010.

[59] B. Karrer, M. E. Newman, Stochastic blockmodels and community structure in networks, Physical review E 83 (1) (2011) 016107.

[60] M. E. Newman, Network structure from rich but noisy data, Nature Physics 14 (6) (2018) 542–545.

[61] L. Song, A. Smola, A. Gretton, K. M. Borgwardt, J. Bedo, Supervised feature selection via dependence estimation, in: International Conference on Machine Learning, 2007, pp. 823–830.

[62] X. Wang, M. Zhu, D. Bo, P. Cui, C. Shi, J. Pei, Am-gcn: Adaptive multi-channel graph convolutional networks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2020, pp. 1243–1253.

[63] D. Niu, J. G. Dy, M. I. Jordan, Multiple non-redundant spectral clustering views, in: International Conference on Machine Learning, 2010, pp. 831–838.

[64] M. Qu, Y. Bengio, J. Tang, Gmnn: Graph markov neural networks, in: International conference on machine learning, PMLR, 2019, pp. 5241–5250.

[65] R. M. Neal, G. E. Hinton, A view of the em algorithm that justifies incremental, sparse, and other variants, in: Learning in graphical models, Springer, 1998, pp. 355–368.

[66] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likelihood from incomplete data via the em algorithm, Journal of the Royal Statistical Society: Series B (Methodological) 39 (1) (1977) 1–22.

[67] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. Ver Steeg, A. Galstyan, Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing, in: International Conference on Machine Learning, 2019, pp. 21–29.

[68] Y. Chen, L. Wu, M. Zaki, Iterative deep graph learning for graph neural networks: Better and robust node embeddings, Advances in Neural Information Processing Systems 33 (2020) 19314–19326.

[69] T. Wu, H. Ren, P. Li, J. Leskovec, Graph information bottleneck, Advances in Neural Information Processing Systems 33 (2020) 20437–20448.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: