This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2024.3355089

Adaptive attention-based graph representation learning to detect phishing accounts on the Ethereum blockchain

Haojie Sun, Zhaowei Liu, *Member, IEEE*, Shenqiang Wang, and Haiyang Wang

***Abstract*—With Ethereum blockchain advancement, the Ethereum platform gathers numerous users. In this context, traditional phishing appears new fraud methods, resulting in significant losses. Currently, network embedding methods are considered effective solutions in the field of phishing detection. However, investigating existing Ethereum phishing node detection algorithms finds they are not optimal and still face two issues. Firstly, the Ethereum network's topology is unsatisfactory, with nodes exhibiting a long-tail distribution in their degree. Current technologies typically allow high-degree nodes to acquire high-quality embeddings, while low-degree nodes, constrained by limited structure, obtain embeddings of lower quality, significantly impacting the detection accuracy of downstream tasks. Secondly, different features of nodes will suffer losses during the fusion process, resulting in the final learned feature embedding being suboptimal. This paper presents an attention-based graphical learning representation approach (ABGRL) to address these problems. ABGRL extracts different feature information by means of multiple channels, and fuses the different feature information using adaptive attention convolution to select the feature information that has the greatest impact on the downstream task. Then the tail node feature information is enhanced by a self-supervised regression model with robust tail node embedding. Finally, the effectiveness of the proposed model was validated through extensive experiments.**

***Index Terms*—Blockchain, Ethereum, Phishing Fraud Detection, Network Embedding, Graph Neural Networks**

I. Introduction

BLOCKCHAIN [1] is a novel digital currency based on a decentralized network that is not governed by any central authority and originated from Bitcoin. In 2008, an anonymous author proposed the concept of blockchain. With blockchain technology's ongoing development, it is widely used in the cryptocurrency industry [2]. Currently, Bitcoin and Ethereum dominate the cryptocurrency market, with a total market share of nearly 70% [3]. Among blockchain platforms, Ethereum is the most widely used [4], which supports smart contracts using a Turing-complete language.

This work was supported in part by the National Natural Science Foundation of China under Grant 62272405, School and Locality Integration Development Project of Yantai City(2022), the Youth Innovation Science and Technology Support Program of Shandong Provincial under Grant 2021KJ080, the Natural Science Foundation of Shandong Province under Grant ZR2022MF238, Yantai Science and Technology Innovation Development Plan Project under Grant 2022XDRH023. (Corresponding author: Zhaowei Liu.)

Zhaowei Liu, and Haojie Sun are with the School of Computer and Control Engineering, Yantai University, Yantai 264005, China (e-mails: lzw@ytu.edu.cn; sunhaojie@s.ytu.edu.cn)

Haiyang Wang,and Shenqiang Wang are with the Institute of Network Technology (Yantai), Shandong, China(e-mail: wanghaiyang@ict.ac.cn; wsqaahh@foxmail.com)

However, as Ethereum becomes increasingly popular, the Ethereum network has also become a breeding ground for various types of cybercrime [5]. More than 10% of Ethereum accounts have reportedly received various threats [6], including phishing scams, honey traps, money laundering, Ponzi schemes, and so on [7], [8]. Among them, phishing scams are the most harmful type of cybercrime [9], accounting for more than half of all Ethereum crimes [10]. Phishing scams include the unlawful alteration of official websites or contact details by con artists in order to steal their victims' personal information [11]. In most cases, this false information can be widely disseminated through email, advertisements, forums, chat applications, etc., with a great deal of damage and relatively low cost. Compared with traditional phishing scams, phishing scams on Ethereum have different manifestations [12]. Firstly, cash is no longer the target of phishing scams, and cryptocurrencies have become the target. Secondly, since the blockchain is transparent, anyone can access every transaction record on Ethereum, providing a complete dataset for researching different Ethereum users [13]. Thirdly, on Ethereum, phishing scams frequently differ from normal phishing scams,and more methods exist for spreading phishing content. [14].

In the face of the aforementioned security issues, how to quickly and accurately identify phishing accounts within the Ethereum platform has become a popular subject in the area of blockchain security [15]. Applying algorithms for network representation learning to the Ethernet network for phishing detection is currently the most popular approach [16], [17]. This method can be roughly divided into two categories: detection algorithms random walks-base and detection algorithms graph neural networks(GNN)-base [18]. Random walk-based approach to detection is a semi-supervised algorithm used for network representation learning and scalable network feature learning [19]. A key aspect of these algorithms is the strategy of random walks, i.e., how to choose neighbors when selecting them in order to include more topological information. GNN-based algorithms use neural networks to learn representations of nodes in the Ethereum network, in order to conduct phishing detection [20], [21]. Typically, these algorithms use a message-passing protocol, feature aggregation is a crucial stage, where each node gathers feature data from its topological neighbors in each convolution layer [22], [23].

Although these two types of algorithms have made great progress in detecting Ethereum phishing nodes, existing algorithms still face two challenges: Firstly, this work by evaluating and analyzing the ability of GNN that fuses network

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2024.3355089

topology features and node attribute features, found that the current GCN model is suboptimal in fusing different feature information of Ethernet network, there is a certain loss in the feature fusion process, and it is not able to optimally extract the feature information that is the most important for downstream tasks from the different features. Secondly, by analyzing the topological structure of the Ethereum transaction network [24], this work found that the degree of nodes in the Ethereum transaction network exhibits the characteristics of a long-tail distribution [25] (that is, most nodes have few connections, and only a few nodes have rich connections, as shown in Figure shown in 1). However, the performance of most existing embedding-based methods largely depends on rich structural information. Although high-degree nodes connected to enough neighboring nodes can fully extract neighbor information, low-degree nodes are connected to small neighborhoods that may be biassed or underrepresented, which can lead to unsatisfactory model performance.

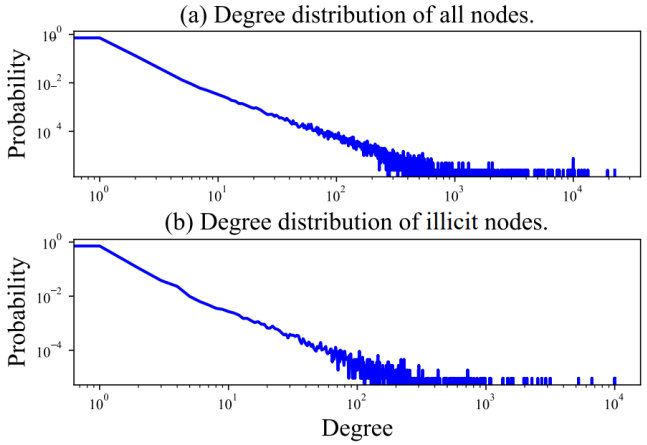

Fig. 1. Degree distribution of nodes.

After finding the inadequacy of the existing Ethereum trading network node detection framework, the question naturally arises: "Can we design a GNN model that can learn effective tail node characteristic vectors from limited structural information and can extract in a better way the implicit relationships between multiple different characteristics, selecting the most important characteristic information that affects downstream tasks?"

Based on the challenges raised above, this work proposes a new network embedding algorithm ABGRL for detecting phishing accounts in the Ethereum network. Specifically, the algorithm first learns different feature information of nodes from the original graph and the node's attribute feature map by using multiple graph representation learning channels for extracting feature information. Then, in order to deal with the problems raised by Challenge 1, the algorithm introduces an adaptive attention mechanism, using an adaptive convolution method to capture the potential relationships between different features and adaptively select the most important feature information for downstream tasks, thus fully ensuring the low-dimensional embeddings contain more effective features that contribute to the downstream task. To overcome the issues raised in Challenge II, this work proposes a self-supervised regression model for robust tail node embedding by learning its neighborhood representation information from structure-rich head nodes, and then further transferring it to structure-limited tail nodes to enhance its representation so that its embedding contains more structural information. Finally, in order to verify the quality of the learned node features, the final node embedding vectors are input into different classifiers to verify the effectiveness of ABGRL.

This article's main contributions are as follows:

(1) This work proposes a graph representation learning-based Ethernet phishing node detection model (ABGRL) that extracts different feature information through multiple feature extraction channels and employs an adaptive attention mechanism to capture the implicit relationships between different features, and adaptively selects to fuse the feature information that is most important for the downstream task.

(2) This work points out that the degrees of nodes in Ethernet networks exhibit long-tailed distributions, and enhances their representation by employing a self-supervised regression model for robust tail node embedding to learn their neighborhood representations from structurally rich head nodes, which are then further transferred to structurally limited tail nodes.

(3) Numerous experiments were carried out on the Ethereum dataset collected in this article. Experimental findings indicate that the approach suggested in this paper outperforms traditional phishing node identification techniques on the Ethereum network.

II. Related Work

This section summarizes the graph representation learning-based anomalous node detection algorithms and Ethernet phishing scam detection algorithms released in recent years, and describes their principles and features. Both types of algorithms apply network embedding techniques to map high-dimensional networks into lower-dimensional spaces, thereby transforming the anomaly node detection problem into a graph node classification problem [26].

A. Anomaly Node Detection Algorithm Based on Graph Representation Learning

The anomaly node detection algorithm based on graph representation learning is a method used to identify abnormal nodes in a network [27]. This class of algorithms learns the topological structure of the graph and relationships between nodes, mapping nodes from a high-dimensional network to a low-dimensional representation space. This enables the low-dimensional node representation to effectively capture both local and global information in the graph structure, providing an effective means of anomaly detection in complex networks. Currently, graph representation learning-based anomaly node detection algorithms can be broadly categorized into two types: random walk-based algorithms and graph neural network(GNN)-based algorithms.

The core of random walk-based algorithms lies in the strategy of selecting random walks, determining how nodes

choose neighbors to maximize the inclusion of neighborhood information. Classic algorithms such as DeepWalk, Node2vec, and Line are based on random walks. DeepWalk [28], as a pioneering algorithm, utilizes random walks to obtain sequences of node neighborhoods, which are then input into Word2Vec to generate vector representations for nodes. Node2vec [29] builds upon DeepWalk, incorporating a biased random walk strategy controlled by tuning hyperparameters.In recent years, many abnormal node detection algorithms based on random walks have also been released. For example, She et al. [30] proposed a local outlier detection method based on grid random walks, which uses random walks to obtain network information. Stationary distribution vector for the grid infographic. Liu et al. [31] proposed a neighborhood relationship filtering model based on random walk similarity measure to find the best neighbor for each relationship in the heterogeneous network.

The other category, graph neural network(GNN)-based algorithms, primarily leverage deep neural networks to learn non-linear information in the graph. Fundamental graph neural network algorithms such as Graph Convolutional Network (GCN) [32], GraphSAGE [33], Cluster-GCN [34], etc., have demonstrated excellent performance in various downstream tasks such as node classification and link prediction. These algorithms utilize deep learning techniques and the message-passing mechanism among nodes in the network to capture dependencies and aggregate feature information from topological neighbors [35].Additionally, some network algorithms based on attention mechanisms to capture the structure and features of the network have been released recently. For example, Zhang et al. [36] proposed an adaptive structural fingerprint model (ADSF), which ”crosstalks” each other by learning multi-head attention, and is able to deal with complex reality. Particularly useful with world data. He et al. [37] proposed a graph joint attention network (CAT) that can learn representations embedding latent features considered important by joint attention.

While there has been a lot of progress in recent years with random walk-based algorithms and graph neural network(GNN)-based algorithms, such algorithms are not specifically designed for the Ether transaction graph, and thus have some limitations in dealing with the complexity of the Ether network. When directly applying the anomalous node detection algorithm based on graph representation learning to the Ethernet network to extract feature information, many important features related to the phishing node detection task, such as transaction feature information on the edges, may not be extracted effectively. Therefore, the accuracy rate is not high when applying them directly to Ethernet networks for phishing node detection.

B. Ethereum Network Phishing and Fraud Detection Algorithm

In response to the characteristics of the Ethereum transaction graph, many scholars have proposed different algorithms to solve the problem of detecting phishing fraud accounts. For example,Wu et al. [38] proposed Tran2vec, a phishing fraud account detection method based on improved Node2vec. This method applies the network embedding technique on Ethernet for the first time to perform phishing detection by mining transaction records. Wang et al. [39], in addressing the Ethereum network phishing detection problem, further enriched the information obtained from the random walk algorithm by employing heterogeneous network representation learning. Xia et al. [40] proposed a new node re-labeling strategy based on Ethereum transaction attributes, including transaction amount, quantity, and direction. Through new labels, nodes and subgraphs can be differentiated, enabling the simultaneous learning of the structure and attribute features of the Ethereum transaction network. Chen et al. [21] introduced a phishing account detection method based on graph convolutional networks and autoencoders. Wen et al. [41] provided a new method, LBPS, for analyzing transaction records. For the first time in the Ethereum phishing node detection task, this method combines manual feature engineering with transaction record-based methods. Liu et al. [26] proposed an adaptive multi-channel Bayesian graph attention network (AMBGAT) to enhance the Ethereum network structure by using Bayesian estimation, thereby improving the accuracy of phishing node identification. Wang et al. [42] suggested a method applying graph representation learning based on temporal graph attention, which models the interaction between time signals and node features, edge features, and the topology of the graph.

To summarize, the current Ethernet phishing account detection models based on graph representation learning have made great progress, but most of the models assume that the network graph structure is complete, and little attention has been paid to the long-tailed distribution problem that exists in the network itself, and many existing Ethernet phishing node detection algorithms do not take into account the implicit relationship between different features in the process of extracting and fusing different features using a direct summation, which may result in the corruption of some feature information that is important to the downstream task.

III. Framework

This section will provide a detailed overview of the methodology proposed in this work, which primarily consists of three modules: the Data Collection and Network Construction module, the Feature Extraction and Fusion module, and the Feature Enhancement and Phishing Detection module. The Data Collection and Network Construction module is responsible for crawling Ethereum transaction data from the web and constructing a network graph containing both phishing and normal nodes. The Feature Extraction and Feature Fusion module is responsible for extracting the different feature information in the topology and attribute maps, and using an adaptive attention approach to capture the potential relationships between the different features, extracting and fusing the feature information that is most important for the downstream task. Finally, the Feature Enhancement and Phishing Detection module enhances the feature vectors of tail nodes and performs subsequent phishing detection tasks based on the enhanced embedding vectors. The details are illustrated in Figure 2.

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2024.3355089

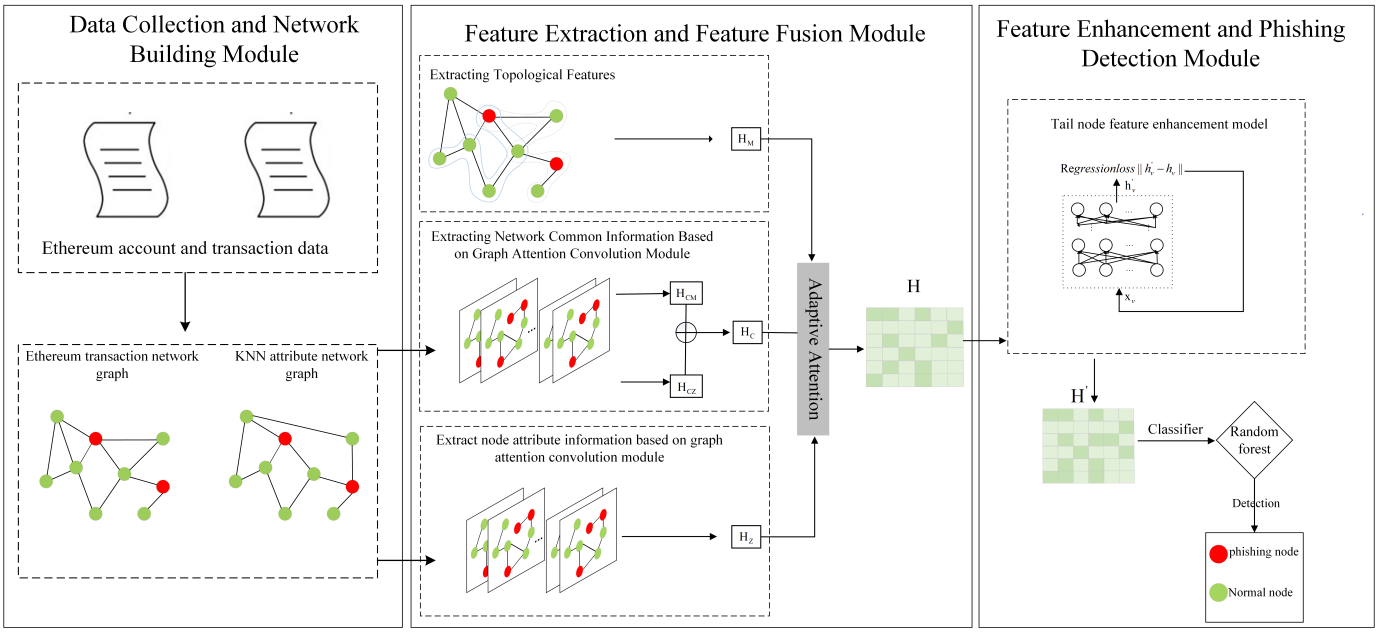

Fig. 2. The overall framework

A. Data Collection and Network Building module

The Ethereum phishing node identification problem studied in this paper can be formulated as a semi-supervised graph node classification problem. In order to achieve this goal, the source and collection of Ethereum transaction network data sets will be described in detail in this sub-section, as well as the process of building transaction networks.

1) Data Collection: To train the proposed model mentioned in this paper, a sufficiently large dataset is needed to support it. Only with an ample amount of training samples can the model exhibit better classification performance. Therefore, the first step is to obtain a comprehensive Ethereum dataset.

Due to the public transparency of Ethereum, all transaction records can be publicly accessed on the blockchain. Thus, by accessing the blockchain network and synchronizing block data using blockchain clients such as Core and Geth, raw Ethereum data can be obtained. In this research, the labeled nodes collected were obtained from a tag cloud module provided by the Etherscan blockchain explorer. This website displays various types of Ethereum account information that have been labeled. In this work, accounts from exchanges, ICO wallets, miners, investment institutions, and other normal types were scraped as normal account nodes, while phishing accounts were scraped as abnormal phishing nodes.

To more effectively identify Ethereum phishing accounts, relying solely on limited labeled data is insufficient. It is necessary to collect a large amount of unlabeled account data to build a comprehensive Ethereum phishing account dataset. When obtaining unlabeled accounts, this work analyzes the recent activity dates of each node and randomly extracts Ethereum accounts based on these dates. To avoid duplicate accounts, the extraction process checks whether the account has been labeled by the Etherscan website, and if so, the address is removed. After obtaining node label information, the Ethereum accounts' first-order transaction data for all nodes is retrieved using the application programming interface (API) provided by Etherscan. The final dataset consists of 376,759 account nodes and 1,048,576 transactions, with each account corresponding to a unique address.

Due to the anonymity of Ethereum accounts, they contain only a small amount of attribute information. To better describe the characteristics of different account nodes, this work extracts account features for each Ethereum account based on the account attribute information and transaction data provided in Etherscan, as shown in Table 1. These features can further reveal correlations between transaction behavior and accounts, enabling the discovery of specific transaction patterns for phishing accounts and providing more data support for Ethereum phishing account identification. The description and calculation methods of some of the attribute characteristics are as follows:

The timestamp of the largest transaction: It is a feature on the transaction edge and represents the timestamp information of the latest transaction between two nodes.

Total amount of transactions: It is a feature on the transaction edge, indicating the total amount of ETH that occurred between the two nodes.

Number of transactions: It is a feature on the transaction edge, indicating the total number of transactions between two nodes.

Total Ether sent: It is an attribute characteristic of the node, indicating the total number of Ether sent by the account as the transaction sender.

Average amount of Ether sent: It is the attribute characteristic of the node and is the value of the total number of Ether sent divided by the number of transactions sent.

2) Network Construction: After collecting the data set, our task is to model the data according to the graph network modeling method based on the collected data set. Model

TABLE I
COMPLETE LIST OF EXTRACTED FEATURES

	Feature
1	The timestamp of the largest transaction
2	The timestamp of the smallest transaction
3	Total amount of the transactions
4	Number of transactions
5	The total gas fee for the transaction
6	Total Ether sent
7	Maximum amount of Ether sent
8	Minimum amount of Ether sent
9	The number of trades accepted as a trade sender
10	Average amount of Ether sent
11	Total Ether received
12	Maximum amount of Ether received
13	Minimum amount of Ether received
14	The number of trades accepted as a trade taker
15	Average amount of Ether received
16	Ratio of Ether sent to received
17	ratio of transaction sender to receiver numbers
18	Average Ether traded
19	The timestamp of the account's last transaction
20	The timestamp of the account's first transaction
21	Total Ethereum Transaction Fees
22	Average Ethereum Transaction Fees

Ethereum's transaction dataset as a graph $G = (V, E, X, A)$, where $V = [\mathrm{V}_1,\ \mathrm{V}_2, \cdots \mathrm{V}_\mathrm{n}]$ is the set of Ethereum network nodes, where the total number of accounts is $N = |V|$, $E = [e_1, e_2, \ldots e_r]$ is the set of r transaction edges, and X is the set of transaction edge attributes with $X \in R^{(r*s)}$, where s is the dimension of the characteristics of the transaction edge and r is the number of transaction edges. A is the set of node attributes with $A \in R^{(n*c)}$, where c is the dimension of the node characteristics and n is the number of nodes.

B. Feature Extraction and Feature Fusion module

This work creates a model named the ABGRL deep neural network for feature extraction and feature fusion. Figure 4 depicts the model's framework. The model first generates embeddings containing node topology information by using a biased random walk method to mine information from transaction edges and the topology of the graph. Next, to learn node feature information from the original graph and node attribute feature maps, multi-channel attention convolution modules are employed, and automatic learning of the importance weights of the various embeddings created above takes place using an adaptive attention method, in order to better fuse them and improve the embedding ability of nodes,and include more information in the combined embeddings. Finally, in order to enhance the representation ability of the tail node embedding and solve the long-tail distribution problem of the Ethereum transaction network, a self-supervised regression model is adopted to learn its neighborhood representation information from the structurally rich head nodes and further transfer it to the structurally limited tail nodes to enhance their representation ability, so that their embeddings contain more structural information.

This section mainly introduces the feature extraction and feature fusion algorithms proposed by us for the Ethereum transaction network. In the next subsection 3.3, the feature enhancement algorithm of the tail node will be introduced.

1) Topological feature learning based on random walk: There are two main components to the random walk-based feature representation learning method. To create node embeddings including topological information, the initial step is to capture the structural relationships and other information between nodes using various walking strategies. Then, the skip-gram model utilizes the produced node sequence to solve a maximum likelihood optimization problem, enabling the learning of node embeddings. With the help of this method, data analysis and processing can effectively learn the feature information of nodes.

Given a source node u, sample a random walk sequence of length l, and the starting vertex of the sequence is denoted as $c_0 = u$. The vertexes in the random walk sequence within l are represented as c_i, and the sampling strategy for c_i is:

$$P(c_i = x \mid c_{i-1} = u) = \begin{cases} \frac{\pi_{u,z}}{Z} & if(x,v) \in E \\ 0 & else \end{cases} \tag{1}$$

where $\pi_{u,x}$ is the likelihood of a transition occurring between node u and node x, whereas Z is the normalisation factor.

In order to understand the characteristics of the transaction network more comprehensively and extract the topology information of the Ethereum transaction network, this paper considers the transaction information between each pair of nodes, such as transaction amount, transaction timestamp, transaction number,and transaction edge type. A biased random walk strategy based on transaction amount, transaction timestamp and transaction number is designed to effectively extract the topological information of the transaction network and embed it into a low-dimensional vector representation.

Individual Information-Based Random Walk Strategy:

In the transaction timestamp-based random walk method, we consider the transaction information between each pair of nodes and use the transaction timestamp as a record. Since there may be multiple transactions between each pair of nodes and each transaction has its own timestamp. Therefore, in order to take into account the timestamp information of all transactions, this work sums and averages all transaction timestamps between each pair of nodes to obtain the timestamp information between the node pair.

$$T(u,x) = \frac{\sum_{i=1}^{c} T_i(u,x)}{c} \tag{2}$$

where c represents the quantity of transactions between nodes x and u, and $T_i(u,x)$ is the timestamp of the i-th transaction between node x and node u.

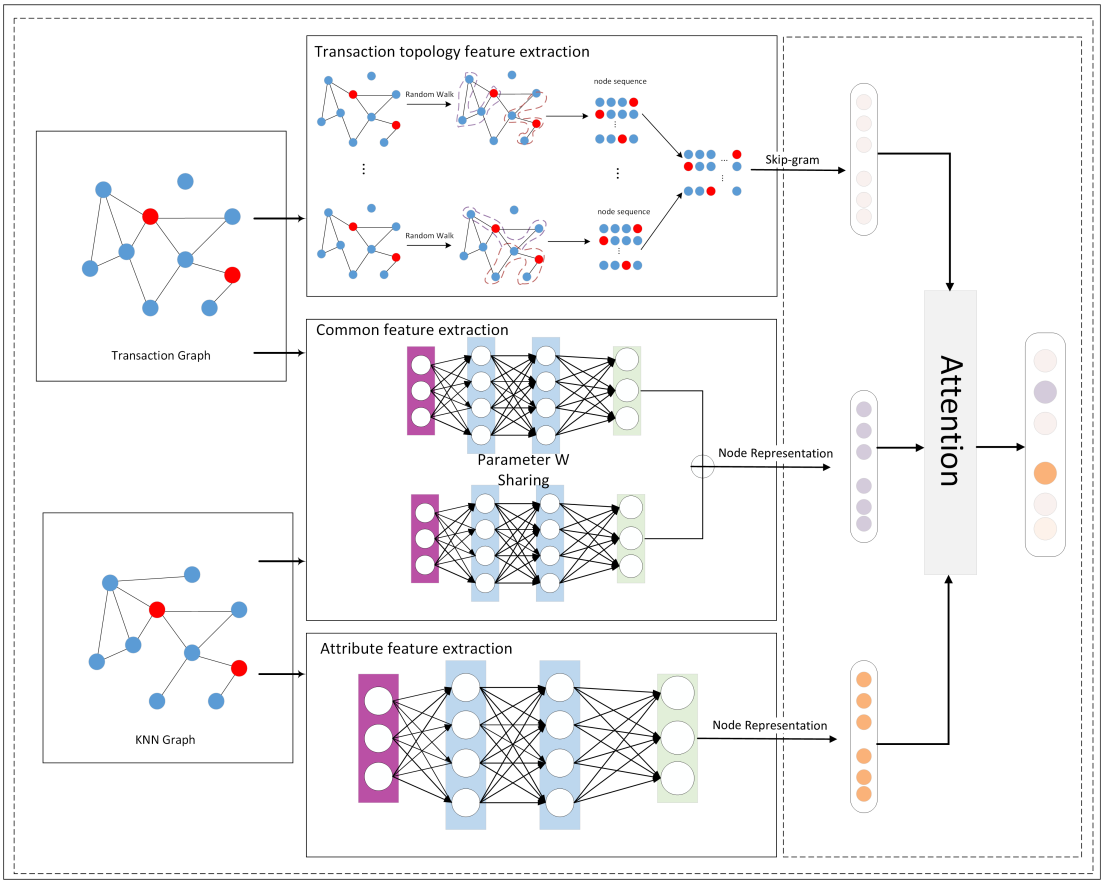

Fig. 3. Framework of ABGRL

Assuming that the greater the timestamp $T(x,u)$ of the obtained transaction, the closer the connection between the two related nodes, our goal is to find the transition probability P between the nodes during the random walk, so that the biased random walk is performed according to the transition probability. Under time-based bias sampling, the transition probability P_T from node u to neighboring node $x \in V_u$ is calculated as follows:

$$P_T = \frac{T(u,x)}{\sum_{x' \in V_u} T(u,x')} \tag{3}$$

Where V_u is represented as a set of nodes directly connected to node u, and $T(u,x)$ refers to the timestamp of node u and node x obtained by formula 2 , $\sum_{x' \in V_u} T(u,x')$ refers to the time stamps of node u and all its adjacent nodes sum of timestamps.

Similarly, under biased sampling based on transaction amount, the transition probability P_A from node u to adjacent node $x \in V_u$ is calculated as follows:

$$P_A = \frac{MAXA(u,x)}{\sum_{x' \in V_u} A(u,x')} \tag{4}$$

Among them, $MAXA(u,x)$ refers to the max transaction amount between node u and node x , $\sum_{x' \in V_u} A(u,x')$ refers to the transaction amount between node u and all its adjacent nodes The sum of transaction amounts.

Similarly, under biased sampling based on the number of transactions, the transition probability P_C from node u to adjacent node $x \in V_u$ is calculated as follows:

$$P_C = \frac{C(u,x)}{\sum_{x' \in V_u} C(u,x')} \tag{5}$$

Where $C(u,x)$ refers to the number of transactions between node u and node x , $\sum_{x' \in V_u} C(u,x')$ refers to the transaction times between node u and all its adjacent nodes The sum of the number of transactions.

Random walk strategy based on various transaction information:

The previous three strategies only considered one type of information on the transaction edges, and therefore, the learned embeddings had some missing information. To ensure that the learned features contain more information, this work simultaneously considers the transaction amount, transaction time, and transaction count information. Using parameters $\alpha, \beta, \gamma \in [1,0]$ to balance the influence of these three types of information on graph feature learning.

Specifically, firstly, normalize the transaction amount, transaction time, and transaction count information separately. $A'_{u,v}$ represents the normalized transaction amount weight information between node u and node v, $T'_{u,v}$' represents the normalized transaction time weight information, and $C'_{u,v}$ represents the normalized transaction count weight information

between node u and node v. Then use the hyperparameters $\alpha, \beta, \gamma \in [1, 0]$ to balance the influence of different weight information.

$$S_{u,v} = \alpha \cdot A'_{u,v} + \beta \cdot T'_{u,v} + \gamma \cdot C'_{u,v} \tag{6}$$

In the updated user transaction network, nodes with higher weights represent stronger relationships and have a greater impact on the central node. By utilizing the aggregation information weight in the transaction graph, it can be transformed into the transition probability between two nodes. Based on the different transition probabilities, when performing random walks from a certain node, neighboring nodes can be selectively chosen. Let the transition probability from node u to its first-order neighboring node v be:

$$P_S(u, v) = MAXS_{u,v} / \sum_{v' \in V_u} S_{u,v'} \tag{7}$$

Here, $MAXS_{u,v}$ refers to the maximum weight between node u and node v after aggregation, and $\sum_{x' \in V_u} S_{u,v'}$ represents the sum of aggregated information weights between node u and all nodes it has transacted with.

To capture the topological features of the Ethereum transaction network graph and embed more transaction information into the representation vectors, perform biased random walks with a step size of l for r iterations on each node u in the network. By taking into account different transition probabilities between nodes, we sample biasedly from the neighbor set N_u of the central node, generating a sequence of nodes containing the topological features of the network. Similar to previous work [29], utilize the skip-gram model and stochastic gradient descent to optimize the node embeddings and obtain the objective function f, which maximizes the probability of the logarithm of node occurrences in the neighborhood N_u of node u. In other words, our goal is to increase the possibility of monitoring the context nodes given the center node.

$$MAX \sum_{u \in V} \log P_r(N_u \mid f(u)) \tag{8}$$

$$f : G_M \to R^{n*d} \tag{9}$$

To learn the topological embedding vector $H_M \in R^{n*d}$ containing the topological information of each node,where n is the quantity of nodes and d is the dimension of the topological embedding of each node.

2) Multi-channel convolution module: This section introduces the convolution module based on the adaptive attention we use. Unlike traditional GCN, without knowing the full graph network beforehand, this module is able to assign various correlation weights, or alpha, to various nodes in the neighborhood based on their relationships. This paper designed two convolution modules, one for extracting node attribute information and the other for extracting common information in different networks. When calculating alpha, the detailed explanation of the formula is as follows:

$$\alpha_{i,j} = \frac{exp(LeakyReLU(\vec{a} \cdot (W^l h_i^l \mid\mid w^l h_j^l)))}{\sum_{k \in N_i} exp(LeakyReLU(\vec{a} \cdot (W^l h_i^l \mid\mid W^l h_k^l)))} \tag{10}$$

Where $\vec{a}$ is the parameter vector of the forward propagation layer, w^l is the weight matrix of each node's linear change, and $||$ represents vector splicing.

Attention convolution module for extracting attribute information:

Create a k-nearest neighbour (KNN) graph in this work based on the node feature matrix X to capture the attribute feature information embedded in the nodes in the feature space $G_f = (A_f, X)$, where A_f is the adjacency matrix of the KNN graph to represent the attributes between nodes similarity. When constructing a KNN graph, first calculate the similarity matrix u by calculating the cosine similarity of attribute information between nodes, and the calculation expression of the similarity matrix is:

$$u_{i,j} = \frac{x_i \cdot x_j}{\mid x_i \mid \cdot \mid x_j \mid} \tag{11}$$

Among them, x_i and x_j are the attribute characteristics corresponding to node i and node j respectively. Then uniformly select the top k similar nodes for each node based on the generated similarity matrix u to build connection edges to generate a KNN graph.

Finally, we take the generated KNN proximity graph G_f as the input graph and input it into the graph convolutional network (GAT) to learn the attribute feature embedding of each node. The output of the last layer of this module is:

$$h_i = \sigma(\frac{1}{k} \sum_{k=1}^{K} \sum_{j \in N_i} \alpha_{i,j} W^k h_j) \tag{12}$$

Where $\alpha_{i,j}$ is the attention coefficient after the kth normalization, W^k is the weight matrix, σ is the activation function, N_i is the neighborhood node set of node i, h_j is the eigenvector of node i's neighbor node j, k is the number of convolutional layers. The output vector of the last layer is represented as H_Z as the attribute feature vector of the node.

Attention convolution module for extracting common information in the network:

In the Ethereum transaction network, the attribute feature space of nodes and the topological space of nodes are not completely independent. To more effectively extract network features, in addition to embeddings that extract node-specific information in one space, embeddings that extract shared information across these two spaces are also required. This is because it can be difficult to determine which part of the original topology graph, the attribute feature map of the nodes, or the common part of the two, is more relevant to the target downstream task. Therefore,this paper create a convolution module to extract the shared data between the initial topology graph and the attribute feature graph of nodes, making the task more flexible.

Firstly, exploit the common convolution module to extract the embedding H_{CM} of nodes from the original topology graph as follows:

$$h_i = \sigma(\frac{1}{k} \sum_{k=1}^{K} \sum_{j \in N_i} \alpha_{i,j} W_c^k h_j) \tag{13}$$

The $\alpha_{i,j}$ coefficients are the normalized attention coefficients after the kth iteration, σ is the activation function, and

W_c^k is the trainable weight matrix in the common convolution module. It is used to extract common information in two feature spaces. The output vector of the last layer is taken as the learned feature and represented by H_{CM}.

Next, input the KNN adjacency graph to learn its features using the same formula as formula 14, and the output vector of the last layer is taken as the learned feature and represented by H_{CZ} .With two different input graphs, we obtain two output embeddings H_{CM} and H_{CZ} . The common information embedding learned on these two feature spaces is expressed as follows:

$$H_C = (H_{CM} + H_{CZ})/2 \tag{14}$$

Adaptive feature fusion module:

After extracting features from the Ethereum transaction network can obtained three types of embedding vectors containing different information, namely the embedding H_M containing topological features, the embedding H_Z containing node attribute features, and the embedding H_C containing common features. Considering that the label information of Ethereum nodes may be related to one or several of these feature information, in order to better integrate these three parts of information and extract the information related to Ethereum node labels, by used adaptive node feature fusion technology to automatically select the information in these three types of features that is more important for downstream tasks, and generate the final node feature embedding.

The fusion process uses an attention-based adaptive mechanism to automatically learn the importance of different embedding information for the task of identifying phishing nodes in Ethereum, that is, for the feature information $[H_M, H_Z, H_C]$ to learn the importance coefficient $[q_m, q_z, q_c] \in R^{n*1}$. Specifically, taking node i as an example, for its feature vector, first apply a nonlinear transformation, and then multiply it by a shared attention vector to get its attention values $[q_m^i, q_z^i, q_c^i] \in R^{n*1}$.

$$q_x^i = \omega^T \cdot tanh(W \cdot (h_x^i)^T + b) \tag{15}$$

where $h_x^i \in R^{1\times d}$ is one of the three feature embeddings of node i, $W \in R^{d'\times d}$ is a trainable weight matrix, $b \in R^{d'\times 1}$ is the paranoid parameter, $\omega \in R^{d'\times 1}$.

Next, normalize the attention values q_m, q_z, q_c using the softmax algorithm to get the final weights:

$$q_m^i = softmax(q_m^i) = \frac{exp(q_m^i)}{exp(q_m^i) + exp(q_z^i) + exp(q_c^i)} \tag{16}$$

The final feature vector is created by joining the learned three weight coefficients with the pertinent feature data:

$$H^i = q_m^i \cdot h_m^i + q_z^i \cdot h_z^i + q_c^i \cdot h_c^i \tag{17}$$

C. Feature Enhancement module

The degree of Ethereum's nodes is distributed in a long tail, and its transaction network is not a perfect network structure. For this kind of network structure, it is not a problem to learn the embedding of the head node with a higher degree, but not for the tail node. The lack of available structural information for tail node embeddings presents the biggest difficulty in learning them. In other words, each tail node has only a small number of neighbor nodes to observe. In this section will use a self-supervised regression model to reconstruct the embedding of the tail node by using the high-quality embedding of the head node to provide it with more topological information without assuming additional auxiliary information.

The precise processing procedure is as follows: classify the graph's nodes into head and tail nodes based on degree, designating the node with a degree greater than 5 as the head node and the node with a degree less than 5 as the tail node. According to the embedding vector generated in Section 3.2.3, separating the head and tail nodes' embedding vectors allows us to train a self-supervised regression model F using the head node's embedding vector as the training set. The regression model F is through continuous Input the training set consisting of the aggregate embedding of the neighbors of the head node to regress and reconstruct the embedding of the generated head node, so that the regression model can still return a good embedding ability when there are only a few neighbor nodes. Specifically, the model reconstructs and outputs a predicted embedding h'_u through the aggregation of the embedding vectors of the neighbor nodes of the given head node, and then trains by continuously optimizing the predicted embedding h'_u to approximate the embedding h_u of the head node. Formally, by reducing the loss error can learn the model's parameters. The processing formula is:

$$L = argmin\sum_{u\in V_{head}} || h'_u - h_u ||^2 \tag{18}$$

The notation $|| \cdot ||$ refers to the Euclidean norm. h'_u represents the reconstructed embedding outputted by the model, while h_u is the original embedding of the head node.

In order to ensure better performance of the trained regression model and make the head node more similar to the tail node, we will filter the neighbors of the head node to simulate the tail node by adjusting the number of neighbors to be similar to the tail node. This allows the model to predict high-quality node embeddings even when only the tail node is available. After training, we can use the regression model F to predict the embedding vector of the tail node, ensuring that its quality can approach that of the structurally rich embedding vector of the head node. The formula for the calculation is:

$$h'_u = W_2 \cdot \sigma(W_1 x_u + b_1) + b_2 \tag{19}$$

Here, σ is an activation function, W_1 and W_2 are weight matrices, and b_1 and b_2 are bias functions. x_u represents the neighborhood features of a node, which are aggregated from its neighbors.

$$N_u^{(m)} = \bigcup_{i\in N_u^{(m-1)}} N_i \tag{20}$$

After obtaining all the m-hop neighbors of the head node, in this work will filter the head node's neighbors. Specifically, only randomly sample k neighbors for each head node, that is, randomly sample k nodes from $N_u^{(m)}$ to form a set N'_u.

$$x_u = AGGR(\{h_i : i \in N'_u\}) \tag{21}$$

AGGR $(\cdot)$ is an aggregator, such as average pooling, LSTM, and graph convolution aggregator.

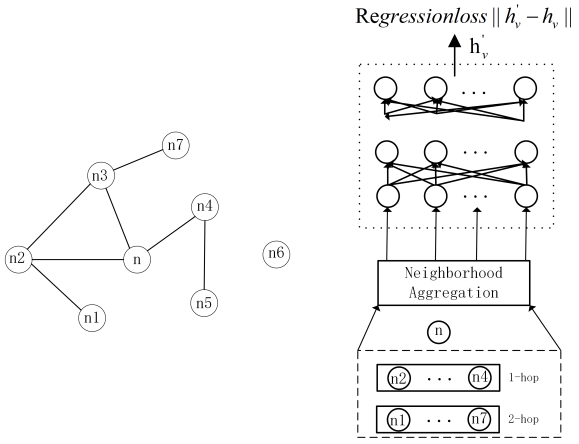

Fig. 4. Training a regression model to reconstruct the embedding of tail nodes

IV. Experiment

In this section, the effectiveness of the proposed ABGRL algorithm will be evaluated using a collected Ethereum dataset. Specifically, first discuss the experimental setup, including the dataset used, baseline methods, and implementation details. Next, a comparison will be made between ABGRL and existing baseline methods to validate the advantages of ABGRL in the task of phishing node identification. Subsequently, an analysis of various variants of ABGRL will be conducted to validate the roles of its different functional modules. Finally, the impact of parameter sensitivity on the phishing node detection task will be analyzed.

A. Experimental Settings

1) Dataset: The dataset used in the experiments comprises 376,759 account nodes and 1,048,576 transactions. Among these data, a total of 9,986 nodes are labeled, with 5,000 identified as normal nodes and 4,956 flagged as phishing nodes.

To comprehensively evaluate the proposed methodology in this study, three different dataset partitioning methods were employed. The details are presented in Table 2, where the dataset is divided into three subsets, namely D1, D2, and D3 [43].

2) Baselines: By analyzing the comparison of baseline methods with similar work, this work compares ABGRL with several different types of algorithms, including random walk-based algorithms, graph neural network-based algorithms, and algorithms specifically applied to Ethereum phishing node detection.

•DeepWalk [28]: is a traditional network embedding algorithm that learns vector representations of nodes by collecting context information and co-occurrence relationships between nodes through random walks.

TABLE II
Three training-validation-test set partition methods

Dataset	Train Set	Validation Set	Test Set
D1	50%	10%	40%
D2	70%	10%	20%
D3	80%	10%	10%

•Node2vec [29]: is based on DeepWalk, obtain node context combinations through biased random walk sampling, and then models this combination to obtain embeddings representing the features of network nodes.

•LINE [44]: can obtain two representation vectors of vertices, source vector and target vector, by optimizing first-order similarity and second-order similarity, and combines the two vectors to represent the final vertex.

•GCN [32]: is a deep learning model for node classification. It applies the neural network's convolution operation to graph structure in general and can process complex graph data, including nodes and edges.

•GAT [45]: is a representative graph convolutional network that adds an attention strategy to produce better neighbor aggregation and also lends some interpretability to the model.

•GEN [46]: is a GNN built on learning graph structures. By constructing an estimated graph and utilizing comprehensive data to estimate a more precise graph structure to adapts the GNN mechanism.

•AM-GCN [47]: is a multi-view graph convolutional network capable of discovering node embeddings from both node features and topology.

•Trans2vec [38]: is a method that detects blockchain network phishing scams by mining blockchain transaction records, considering both transaction amounts and timestamps.

•HNRL [26]: is a heterogeneous network representation learning method that mines hidden information in Ethereum transactions and maps the Ethereum transaction network to a low-dimensional space.

3) Implementation Details: Regarding implementation details, the implementation of ABGRL consists of a random walk module based on multiple transaction data and two GAT layers with the same three hidden layer dimensions and output layer dimensions. The settings in the random walk module are: walk length $l = 10$, number of walks $r = 10$, neighbor window size $m = 5$, weight parameters $\alpha, \beta, \gamma \in [0, 1]$, output embedding dimension $d = 128$. Each layer in the GAT layer contains $k = 4$ attention head calculations, the learning rate is 0.01, the weight decay is 5e-4, the decay rate of each layer is 50%, and the Adam optimizer is used in training. In addition, the dimension of the hidden layer embedding is selected from $\{512, 768\}$, the dimension of the output layer is positioned at 128, the feature map $k = 5$, the number of iterations of the model is set to 100, and the parameters with the highest verification accuracy are saved for testing. In the comparative experimental parameter settings, the random walk-based algorithm settings are the same as the random

TABLE III
NODE CLASSIFICATION RESULTS OF DIFFERENT METHODS

Method	Dataset	D1			D2			D3		
	Metric	Pre	Recall	F1	Pre	Recall	F1	Pre	Recall	F1
Random Walk	DeepWalk	72.00	72.15	72.00	73.32	73.36	73.31	71.45	71.66	71.45
	Node2vec	75.54	75.67	75.53	76.29	76.36	76.28	75.31	75.43	75.30
	Line	76.63	76.39	75.33	77.82	77.36	76.48	78.12	78.32	78.11
Deep Leaning	GCN	76.18	75.07	74.27	76.94	75.63	74.99	79.41	77.85	78.16
	GAT	76.45	73.50	76.50	75.37	70.43	75.33	77.85	71.74	77.71
	GEN	83.31	86.58	84.92	84.76	86.69	85.71	84.83	87.23	86.01
	AM-GCN	79.79	79.66	79.70	82.65	82.31	82.45	80.94	80.09	80.39
Blockchain method	Trans2vec	81.80	81.66	81.70	82.40	82.42	82.41	83.71	83.72	83.71
	HNRL	86.60	89.10	89.00	88.70	91.50	91.70	88.90	**95.90**	**95.70**
Ours	ABGRL	**89.55**	**89.42**	**91.49**	**91.69**	**91.57**	**91.63**	**92.39**	92.22	92.33

walk module. The walk length $l = 10$, the number of walks $r = 10$, the neighbor window size $m = 5$, the paranoia parameter $q = 0.5$, and the output embedding dimension $d = 128$. The algorithm settings based on the graph neural network are the same as the GAT layer. The dimension of the hidden layer embedding is selected from $\{512, 768\}$, and the dimension of the output layer is positioned at 128. The basic settings of trans2vec are the same as the random walk module, except that its importance parameters $p = 0.25$, $q = 0.75$, search bias $\alpha = 0.5$. The edge dimension in HNRL is set e=15,base embedding dimension b=200, attribute dimension a=15, overall dimension d=128, and search strategy parameter q=0.25.

In order to evaluate the classification effects of different algorithms, this article selects three evaluation indicators: Precision, Recall and F1-score. These three indicators are calculated as follows:

$$Pre = \frac{TP}{TP + FP} \tag{22}$$

$$Recall = \frac{TP}{TP + FN} \tag{23}$$

$$F1 = 2 \times \frac{Pre \times Recall}{Pre + Recall} \tag{24}$$

Among them, TP means the prediction is positive and the actual value is positive; FP means the prediction is positive and the actual value is negative; FN means the prediction is negative and the actual value is positive; TN means the prediction is negative and the actual value is also negative.

B. Classification experiment results and analysis

Based on the above parameter settings, this chapter evaluates the performance of ABGRL in node classification tasks on the Ethereum network. The results are shown in Table 3.

Combined with the experimental results in Table 3, the following conclusions can be drawn:

(1)The performance of the random walk-based approach ranges between 72% and 79%, and the performance of the deep learning-based approach ranges between 76% and 86%, which is significantly lower than the algorithms that specifically deal with Ethernet phishing node detection. This result proves that traditional graph representation learning algorithms do face certain limitations when dealing with Ethernet networks. When extracting features of the Ethernet transaction network, these traditional algorithms may not be able to effectively capture many important feature information relevant to the phishing node detection task, resulting in a low accuracy rate.

(2)ABGRL exhibits significant node classification advantages over classical graph representation learning algorithms such as DeepWalk, Node2Vec and GCN. This observation highlights the superior performance of ABGRL in capturing complex feature relationships in Ethernet transaction graph data in comparison to traditional graph representation learning methods in Ethernet networks, enabling it to accurately extract the implicit information between different features, extract the feature information whose fusion has the greatest impact on the downstream tasks, and significantly improve the accuracy of the Ethernet fishing node identification task.

(3)Compared with Trans2vec and HNRL, the algorithms specifically designed for Ethereum phishing node detection, ABGRL's accuracy on the three data sets is better than them, ranging from 0.1% to 8% higher. This shows that compared with advanced phishing node detection algorithms, ABGRL can more effectively extract important feature information related to phishing node detection tasks in the Ethereum network, and enhance the feature representation capabilities of tail nodes in the network in a more robust way, thereby improving node classification performance.

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2024.3355089

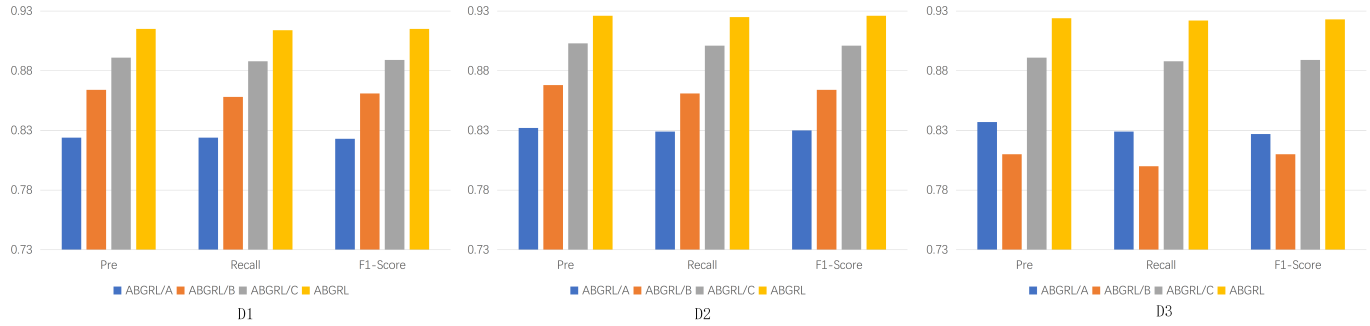

Fig. 5. The results of ABGRL and its three variants on the Ethereum transaction dataset.

C. Ablative Analysis

In order to validate the effectiveness of different modules in ABGRL, this paper compares ABGRL with its three variants on the Ethernet dataset to validate the reasonableness and effectiveness of the proposed model in this paper. Among them, ABGRL/A extracts feature information from the graph using only the random wandering module based on multiple transaction data for detecting phishing nodes.ABGRL/B fuses only topological features and node attribute features generated by random wandering in the graph attention fusion module without fusing the shared features.ABGRL/C is a model that removes the tail node feature enhancement module based on ABGRL .

According to the corresponding observations shown in Figure 5, the results are as follows:

(1) The performance of ABGRL/A significantly decreased compared to ABGRL, with a decrease of 9%, 8.5%, and 9.8% in accuracy on the D1, D2, and D3 datasets, respectively. This result indicates that the features learned solely through random walks are not enough to accurately identify phishing nodes in Ethereum phishing detection tasks. It is necessary to fully learn and integrate various features to achieve better results.

(2) The performance of ABGRL/B also significantly decreased compared to ABGRL. This result indicates that learning only the respective features of the two graph structures is not enough, and it is necessary to learn and integrate the common information of the two graph structures to better identify phishing nodes.

(3) After removing the tail node feature enhancement module, the performance of ABGRL/C was not as good as that of ABGRL. This suggests that the long-tail distribution of nodes in the Ethereum transaction network that we analyzed does indeed affect the performance of algorithm, and solving this problem is crucial for improving the accuracy of Ethereum phishing node detection tasks.

By comparing ABGRL with its variants, the contribution of different modules to the overall performance of the model is revealed, further verifying the effectiveness of the overall model.

D. Parameter sensitivity analysis

There are numerous variables for the proposed method that can impact the outcomes. In this section, we evaluate the influence of several parameters on how well ABGRL performs when tasked with categorizing nodes on the Ethereum transaction network. When a particular parameter is evaluated, the other parameters are set to their default values. Firstly, we evaluate the impact of node embedding dimension on classification performance. When the node embedding dimension is equal to 8, 16, 32, 64, 128, 256, we test the classification effect of ABGRL. Figure 6 depicts the ultimate classification outcome. Figure 6 demonstrates that as the embedding dimension increases, the classification performance of the framework proposed in this paper improves. This shows that the larger the dimension of the node vector, the richer the network structure and node information can be preserved. When high-precision embedding results are required, larger embedding dimensions should be chosen.

Then analyze the influence of the hyperparameters $l, r, \alpha, \beta, \gamma$ of the random walk module and the knn proximity graph K in the attention fusion module. Through a series of experiments, the conclusions that can be drawn are as follows: (1) The values of parameters l and r are 20 and 5, respectively, and the effect is the best. Compared with parameter r, the parameter l has a greater impact on the performance of the random walk module. (2) When the hyperparameters α, β and γ are respectively (0.1, 0.05, 0.1), the model has the best performance. At this time, the information learned by the random walk module is the most abundant. (3) Adjust the k of the knn graph between 2 and 10, and the results show that when k is equal to 5, the K-adjacent graph at this time will contain more node attribute information.

V. Conclusion

This work proposes an Ethereum network representation learning method to extract the characteristics of each node from the Ethereum transaction network to handle the task of phishing node detection. Specifically, the Ethereum transaction network is first built through the collected data. Then, a semi-supervised self-attention network convolution method is proposed, which fuses the obtained network topology embedding and node attribute embedding to enhance the feature information of nodes. The topology embedding of the network is through a random walk proposed in this article. strategies to learn. Finally, a self-supervised regression model is adopted to enhance the expressive ability of tail node embeddings. Experimental results show that the algorithm proposed in this

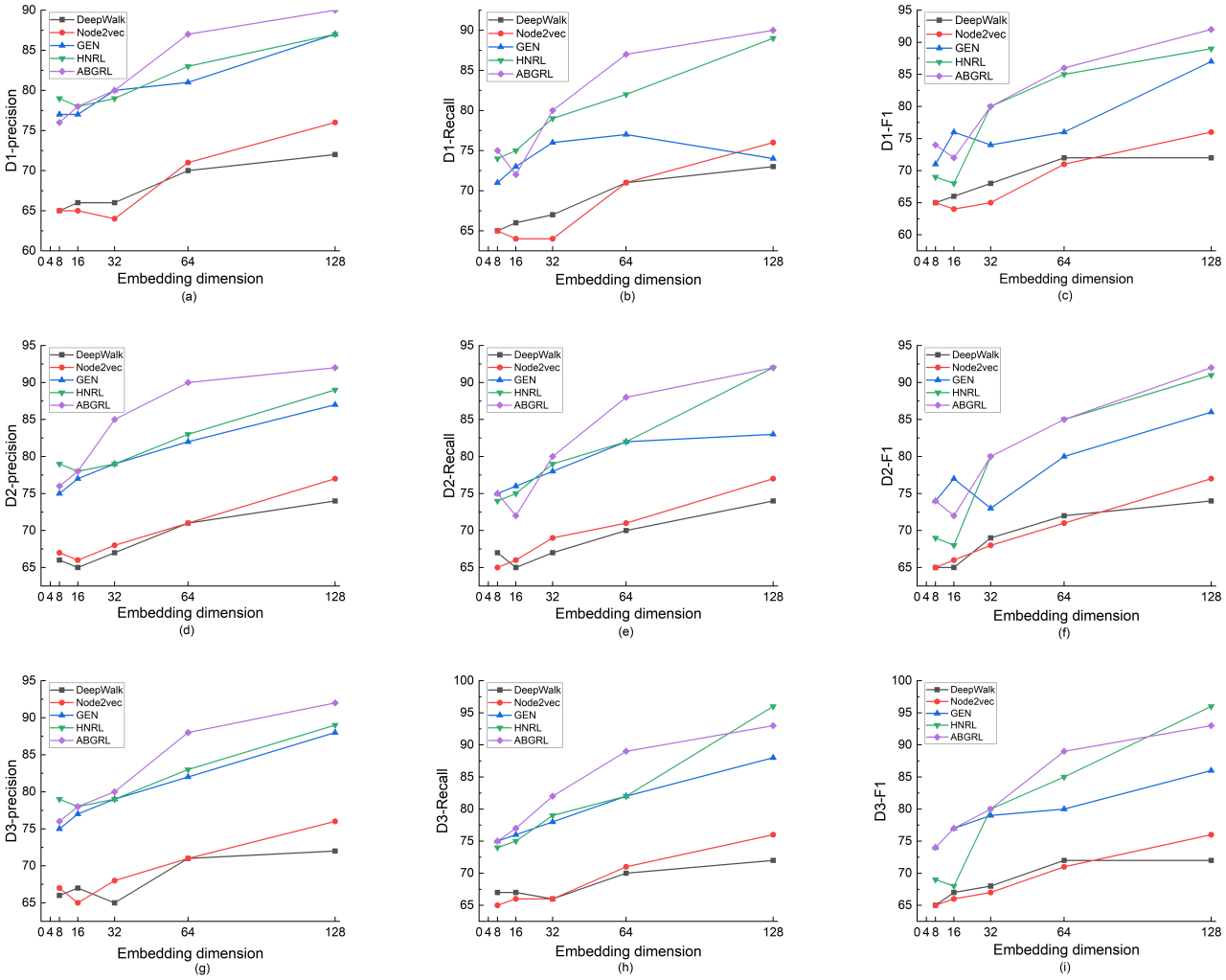

Fig. 6. Impact of Embedded Dimensions on Metrics.

article is better than the existing Ethereum abnormal node detection algorithm. However, in actual application scenarios, the transaction network of Ethereum is dynamic. In future work, the identification of phishing nodes in the dynamic Ethereum network should be studied more deeply.

REFERENCES

[1] S. Nakamoto and A. Bitcoin, "A peer-to-peer electronic cash system," *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, vol. 4, no. 2, 2008.

[2] Z. Zheng, W. Chen, Z. Zhong, Z. Chen, and Y. Lu, "Securing the ethereum from smart ponzi schemes: Identification using static features," *ACM Transactions on Software Engineering and Methodology*, 2022.

[3] M. Vasek and T. Moore, "There's no free lunch, even using bitcoin: Tracking the popularity and profits of virtual currency scams," in *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers 19*. Springer, 2015, pp. 44–61.

[4] Z. Yuan, Q. Yuan, and J. Wu, "Phishing detection on ethereum via learning representation of transaction subgraphs," in *Blockchain and Trustworthy Systems: Second International Conference, BlockSys 2020, Dali, China, August 6–7, 2020, Revised Selected Papers 2*. Springer, 2020, pp. 178–191.

[5] J. Wu, J. Liu, W. Chen, H. Huang, Z. Zheng, and Y. Zhang, "Detecting mixing services via mining bitcoin transaction network with hybrid motifs," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 4, pp. 2237–2249, 2021.

[6] M. Russon, "Ethereum under siege: Scammers make 700000 in 6 days from slack and reddit phishing attacks," 2017.

[7] X. Liu, Z. Tang, P. Li, S. Guo, X. Fan, and J. Zhang, "A graph learning based approach for identity inference in dapp platform blockchain," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 1, pp. 438–449, 2020.

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

[9] Z. Liu, D. Yang, S. Wang, and H. Su, "Adaptive multi-channel bayesian graph attention network for iot transaction security," *Digital Communications and Networks*, 2022.

[10] A. Lazarenko and S. Avdoshin, "Financial risks of the blockchain industry: A survey of cyberattacks," in *Proceedings of the Future Technologies Conference (FTC) 2018: Volume 2*. Springer, 2019, pp. 368–384.

[11] C. F. Torres, M. Steichen, and R. State, "The art of the scam: Demystifying honeypots in ethereum smart contracts," *arXiv preprint arXiv:1902.06976*, 2019.

[12] B. B. Gupta, N. A. Arachchilage, and K. E. Psannis, "Defending against phishing attacks: taxonomy of methods, current issues and future directions," *Telecommunication Systems*, vol. 67, pp. 247–267, 2018.

[13] A. Almomani, M. Alauthman, M. T. Shatnawi, M. Alweshah, A. Alrosan, W. Alomoush, and B. B. Gupta, "Phishing website detection with semantic features based on machine learning classifiers: A comparative study," *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 18, no. 1, pp. 1–24, 2022.

[14] W. Zheng, Z. Zheng, H.-N. Dai, X. Chen, and P. Zheng, "Xblock-eos: Extracting and exploring blockchain data from eosio," *Information Processing & Management*, vol. 58, no. 3, p. 102477, 2021.

[15] R. Li, Z. Liu, Y. Ma, D. Yang, and S. Sun, "Internet financial fraud detection based on graph learning," *Ieee Transactions on Computational Social Systems*, 2022.

[16] D. Lin, J. Wu, Q. Yuan, and Z. Zheng, "Modeling and understanding ethereum transaction records via a complex network approach," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 11, pp. 2737–2741, 2020.

[17] S. Li, G. Gou, C. Liu, C. Hou, Z. Li, and G. Xiong, "Ttagn: Temporal transaction aggregation graph network for ethereum phishing scams detection," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 661–669.

[18] W. Hou, B. Cui, and R. Li, "Detecting phishing scams on ethereum using graph convolutional networks with conditional random field," in *2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. IEEE, 2022, pp. 1495–1500.

[19] W. Chen, X. Guo, Z. Chen, Z. Zheng, and Y. Lu, "Phishing scam detection on ethereum: Towards financial security for blockchain ecosystem." in *IJCAI*, vol. 7, 2020, pp. 4456–4462.

[20] F. Wu, T. Zhang, and A. H. de Souza Jr, "au2, christopher fifty, tao yu, and kilian q," *Weinberger. Simplifying graph convolutional networks*, vol. 5, 2019.

[21] L. Chen, J. Peng, Y. Liu, J. Li, F. Xie, and Z. Zheng, "Phishing scams detection in ethereum transaction network," *ACM Transactions on Internet Technology (TOIT)*, vol. 21, no. 1, pp. 1–16, 2020.

[22] M. Ndiaye and P. K. Konate, "Cryptocurrency crime: Behaviors of malicious smart contracts in blockchain," in *2021 International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE, 2021, pp. 1–8.

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

[24] J. Li, H. Li, N. Cheng, W. Zhang, Y. Xu, H. Zhang, and J. Li, "Multi-channel walk embedding based ethereum phishing scam detection method," in *2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2023, pp. 289–295.

[25] Z. Liu, W. Zhang, Y. Fang, X. Zhang, and S. C. Hoi, "Towards locality-aware meta-learning of tail node embeddings on networks," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 975–984.

[26] Z. Liu, D. Yang, Y. Wang, M. Lu, and R. Li, "Egnn: Graph structure learning based on evolutionary computation helps more in graph neural networks," *Applied Soft Computing*, p. 110040, 2023.

[27] P. Li, H. Yu, X. Luo, and J. Wu, "Lgm-gnn: A local and global aware memory-based graph neural network for fraud detection," *IEEE Transactions on Big Data*, 2023.

[28] B. Perozzi, R. Al-Rfou, and 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.

[29] A. Grover and 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.

[30] C. She and S. Zeng, "An enhanced local outlier detection using random walk on grid information graph," *The Journal of Supercomputing*, vol. 78, no. 12, pp. 14 530–14 547, 2022.

[31] Z. Liu, Y. Wang, S. Wang, X. Zhao, H. Wang, and H. Yin, "Heterogeneous graphs neural networks based on neighbour relationship filtering," *Expert Systems with Applications*, p. 122489, 2023.

[32] S. Fu, S. Wang, W. Liu, B. Liu, B. Zhou, X. You, Q. Peng, and X.-Y. Jing, "Adaptive graph convolutional collaboration networks for semi-supervised classification," *Information Sciences*, vol. 611, pp. 262–276, 2022.

[33] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[34] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 257–266.

[35] B. Fu, X. Yu, and T. Feng, "Ct-gcn: a phishing identification model for blockchain cryptocurrency transactions," *International Journal of Information Security*, vol. 21, no. 6, pp. 1223–1232, 2022.

[36] K. Zhang, Y. Zhu, J. Wang, and J. Zhang, "Adaptive structural fingerprints for graph attention networks," in *International Conference on Learning Representations*, 2019.

[37] T. He, Y. S. Ong, and L. Bai, "Learning conjoint attentions for graph neural nets," *Advances in Neural Information Processing Systems*, vol. 34, pp. 2641–2653, 2021.

[38] J. Wu, Q. Yuan, D. Lin, W. You, W. Chen, C. Chen, and Z. Zheng, "Who are the phishers? phishing scam detection on ethereum via network embedding," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 2, pp. 1156–1166, 2020.

[39] Y. Wang, Z. Liu, J. Xu, and W. Yan, "Heterogeneous network representation learning approach for ethereum identity identification," *IEEE Transactions on Computational Social Systems*, 2022.

[40] Y. Xia, J. Liu, and J. Wu, "Phishing detection on ethereum via attributed ego-graph embedding," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 5, pp. 2538–2542, 2022.

[41] T. Wen, Y. Xiao, A. Wang, and H. Wang, "A novel hybrid feature fusion model for detecting phishing scam on ethereum using deep neural network," *Expert Systems with Applications*, vol. 211, p. 118463, 2023.

[42] L. Wang, M. Xu, and H. Cheng, "Phishing scams detection via temporal graph attention network in ethereum," *Information Processing & Management*, vol. 60, no. 4, p. 103412, 2023.

[43] J. Liu, J. Zheng, J. Wu, and Z. Zheng, "Fa-gnn: Filter and augment graph neural networks for account classification in ethereum," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 4, pp. 2579–2588, 2022.

[44] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*, 2015, pp. 1067–1077.

[45] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio *et al.*, "Graph attention networks," *stat*, vol. 1050, no. 20, pp. 10–48 550, 2017.

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

[47] X. Wang, M. Zhu, D. Bo, P. Cui, C. Shi, and J. Pei, "Am-gcn: Adaptive multi-channel graph convolutional networks," in *Proceedings of the 26th ACM SIGKDD International conference on knowledge discovery & data mining*, 2020, pp. 1243–1253.