








PEAE-GNN: Phishing Detection on Ethereum via Augmentation Ego-Graph Based on Graph Neural Network

Hexiang Huang , Xuan Zhang , Member, IEEE, Jishu Wang , Member, IEEE, Chen Gao , Graduate Student Member, IEEE, Xue Li , Rui Zhu , Member, IEEE, and Qiuying Ma 

Abstract—Recent years, the successful application of blockchain in cryptocurrency has attracted a lot of attention, but it has also led to a rapid growth of illegal and criminal activities. Phishing scams have become the most serious type of crime in Ethereum. Some existing methods for phishing scams detection have limitations, such as high complexity, poor scalability, and high latency. In this article, we propose a novel framework named phishing detection on Ethereum via augmentation ego-graph based on graph neural network (PEAE-GNN). First, we obtain account labels and transaction records from authoritative websites and extract ego-graphs centered on labeled accounts. Then we propose a feature augmentation strategy based on structure features, transaction features and interaction intensity to augment the node features, so that these features of each ego-graph can be learned. Finally, we present a new graph-level representation, sorting the updated node features in descending order and then taking the mean value of the top n to obtain the graph representation, which can retain key information and reduce the introduction of noise. Extensive experimental results show that PEAE-GNN achieves the best performance on phishing detection tasks. At the same time, our framework has the advantages of lower complexity, better scalability, and higher efficiency, which detects phishing accounts at early stage.

Index Terms—Blockchain, Ethereum, graph classification, graph neural network, phishing detection.

Manuscript received 14 October 2022; revised 28 March 2023; accepted 25 December 2023. This work was supported in part by the Science Foundation of Young and Middle-aged Academic and Technical Leaders of Yunnan under Grant 202205AC160040; in part by the Science Foundation of Yunnan Jinzhi Expert Workstation under Grant 202205AF150006; in part by the Major Project of Yunnan Natural Science Foundation under Grant 202302AE09002003; in part by the Science and Technology Project of Yunnan Power Grid Co., Ltd. under Grant YNKJXM20222254; in part by the Knowledge-driven Smart Energy Science and Technology Innovation Team of Yunnan Provincial Department of Education; in part by the Open Foundation of Yunnan Key Laboratory of Software Engineering under Grant 2023SE101; in part by the 14th Postgraduate Research Innovation Project of Yunnan University under Grant KC-2222958; and in part by the Postgraduate Research and Innovation Foundation of Yunnan University under Grant 2021Y399. (Corresponding author: Xuan Zhang.)

Hexiang Huang, Xue Li, and Qiuying Ma are with the School of Software, Yunnan University, Kunming 650091, China (e-mail: roland@mail.ynu.edu.cn; zhxuan@ynu.edu.cn; shirley@mail.ynu.edu.cn; rzhu@ynu.edu.cn; maqiuying@mail.ynu.edu.cn).

Xuan Zhang and Rui Zhu are with the School of Software, Yunnan University, Kunming 650091, China, and also with the Yunnan Key Laboratory of Software Engineering, Yunnan University, Kunming 650091, China.

Jishu Wang and Chen Gao are with the School of Information Science and Engineering, Yunnan University, Kunming 650091, China (e-mail: cswangjishu@hotmail.com; 929165733@qq.com).

Digital Object Identifier 10.1109/TCSS.2023.3349071

I. INTRODUCTION

IN late 2013, Vitalik Buterin, the founder of Ethereum, released the initial version of the Ethereum white article,¹ gathering a group of developers who share the Ethereum philosophy one after another in the global cryptocurrency community to launch Ethereum project. Afterwards, with its mass adoption, Ethereum became one of the most popular platforms.

Ethereum as a blockchain platform, its fundamental technologies include cryptography, peer-to-peer transmission, consensus mechanism, etc. These technologies provide a decentralized platform for Ethereum cryptocurrency transactions. Decentralization makes the system data untamperable, greater fault tolerance and less attackable, but without a regulatory organization. Meanwhile, blockchain participants are anonymous and can transact without authentication, which is known as privacy protection. Therefore, the anonymity can be used by malicious users to commit cybercrimes, which may cause huge losses. A variety of financial scams have been reported. Based on the statistics of Etherscan's label word cloud,² the known anonymous account categories include gambling, mining, exchange, phishing, hacking, etc., the most common of which is phishing scams [1].

Fig. 1 shows the framework of phishing scams in cryptocurrencies. The fraudsters disguise as trusted official websites, send malicious links, in an attempt to obtain users' sensitive information, such as usernames, passwords, etc. Once the user leaks the password, the fraudsters will simply transfer their cryptocurrency without any rescue measures.

Compared with traditional phishing scams [2], phishing scams in cryptocurrencies have the following characteristics.

- 1) Fraudsters often use high-yield promotions to induce transfers, such as Ponzi schemes [3], [4], while rarely stealing the wallets of users, which makes them more invisible.
- 2) The number of accounts that can be created by any individual or organization is unlimited and it is unnecessary to verify identity when creating an account. Therefore, fraudsters often own more than one accounts, one for the

¹<https://ethereum.org/en/whitepaper/>

²<https://etherscan.io/labelcloud>

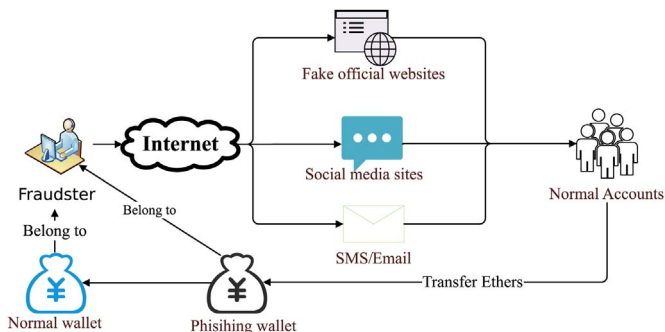


Fig. 1. Framework of phishing scams in cryptocurrencies.

phishing account and another for the external normal account. Fraudsters transfer the digital currency they receive to their normal accounts in stages, after which they can use it normally without any money laundering operations.

- 3) Ethereum phishing scams do not have a fixed pattern or portrait information about the fraudster.

Etherscan³ published 2041 phishing accounts in 2020, rising to 6129 as of this writing in 2022, and the number continues to grow rapidly. Besides, according to the latest phishing scam statistics from blockchain analytics group Chainalysis [5], victims lost \$645 000 in the first week of the phishing campaign, and attackers made more than \$3 000 000 of illicit profits in just one month. Phishing scams cause huge financial losses and have become a major threat to the security of Ethereum transactions [6]. Therefore, it is very urgent to detect phishing accounts in Ethereum for its healthy development [7], [8].

However, the anonymity and no regulatory organization make phishing scams difficult to detect. First, we need adequate verified samples. Second, the accounts are so numerous that the manual checking is inefficient and time-consuming. Then, the nodes in the trading network is in the billions now, and phishing accounts only account for an extremely small fraction. If we use the whole transaction network graph, we are bound to face the storage constraint. Finally, The lifecycle of a phishing account is much shorter than normal accounts, usually less than 20 days [9]. In order to enhance anonymity, fraudsters usually transfer the illegal gains as soon as possible and stop using the original account. Therefore, we need a fast and effective approach to detect phishing scams, which poses new challenges for phishing detection on Ethereum.

Some previous works used traditional machine learning algorithms to identify illegal accounts in blockchains [10], [11], [12]. Since higher-order implicit information is not considered, these approaches have weak classification capabilities. Later, some works based on graph neural networks [13], [14], [15] consider phishing detection as a node classification task, simply extracting transaction amount and transaction times from the transaction records as feature information. Notably, the node classification task usually requires a graph containing all nodes as input. Although these approaches achieve satisfactory performance on phishing detection tasks, they have the disadvantages of high computational complexity, high storage

requirements, poor scalability, and inability to achieve rapid detection of phishing scams at an early stage. To overcome these problems, some work take phishing detection as a graph classification task, through extracting a subgraph of the target account as its representation [16], [17], [18], [19]. But these methods do not take into account finer-grained information, such as the transaction features and interaction intensity. With the development of technology, the detection accuracy still have room for improvement, and it is still a challenging problem to effectively find a limited number of phishing accounts.

To address the above challenges, we propose a framework called PEAE-GNN. We first obtain phishing account labels and transaction data from the authoritative website Etherscan, and then construct K-hop directed ego-graphs with accounts as nodes and transactions as directed edges, respectively. In this article, we turn phishing detection into a graph classification task, only using the local structure topology graph of the target accounts, which reduces the storage pressure. First, through an in-depth analysis of Ethereum phishing accounts, we design a key component for network embedded learning, namely feature augmentation. Specifically, we propose a strategy for node feature augmentation based on structure features, transaction features, and interaction intensity, and using high-order domain information to augment node features. Secondly, we combine the proposed feature augmentation component and deep learning techniques to construct a novel embedding model for Ethereum transaction networks, PEAE-GNN. Finally, we use the mean of top n node features to obtain graph-level representations, which is fed into a classifier to identify phishing scam accounts. We conduct extensive experiments on real-world datasets and the experimental results demonstrate the effectiveness of PEAE-GNN for classifying phishing accounts on Ethereum.

The main contributions of this article are summarized as follows.

- 1) We propose a feature-enhanced phishing detection method for Ethereum transaction networks, improving detection performance by combining features on structure, transaction, and interaction intensity. This method enables efficient phishing detection and can detect phishing scams at an early stage.
- 2) We propose a new graph-level representation for Ethereum phishing detection. Specifically, the updated node features are sorted in descending order, and then the average of the top n nodes is taken to obtain the graph representation. This method can effectively retain key information and reduce introduction of noise.
- 3) Extensive experimental results demonstrate that our method performs well on phishing detection tasks, and outperforms state-of-the-art methods on several metrics. It is worth mentioning that the scalability of our method is good and can be easily applied to account detection of other types.

The rest of this article is organized as follows. First, we summarize the related work on malicious account detection in Section II. Then, we describe the proposed PEAE-GNN framework in detail in Section III. Next, we conduct

³<https://etherscan.io/>

extensive experiments and analyze the experimental results in Section IV. Finally, we conclude the article and outline future work in Section VI.

II. RELATED WORK

Phishing scam detection on blockchain is a new fraud scenario. The existing work is divided into three main categories as follows.

A. Traditional Machine Learning

Chen et al. [20] extracted 219-dimensional statistical features from the first-order and second-order neighbors of nodes, including the node's in-degree, out-degree, and maximum transaction value. Then they used an integrated machine learning algorithm based on LightGBM to identify network phishing nodes. Ibrahim et al. [21] used three different machine learning algorithms: decision tree (j48), random forest, and K-nearest neighbors (KNN) to detect illegal Ethereum accounts. Weber et al. [22] designed various features of transaction timestamps to express the transaction history of an account and constructed a classifier for anomalous bitcoin accounts. Prado-Romero et al. [23] presented a community-based detection algorithm that could get an outlier ranking list to identify the addresses in Bitcoin, which was involved in mixing currency services by setting the outlier of the labeled addresses as the minimum threshold value. Wen et al. [12] first proposed a general phishing detection framework based on feature engineering and then proposed a phishing hiding framework combing the greedy selection mechanism with four phishing hiding strategies to measure the robustness of the proposed general detection models. All of the above methods contribute to the detection of blockchain malicious accounts, but they are characterized by weak classification ability because they do not take into account the implicit information of the blockchain transaction network topology.

B. Graph Embedding

It is well known that during phishing scams, Ether flow for most transactions is gathered from the outside to the central phishing address. From the network perspective, the local topology of phishing nodes may prefer multiple input and single output. In order to learn to characterize the topology of blockchain transaction networks, Wu et al. [24] proposed a graph random walk method called trans2vec to extract account features for phishing detection on Ethereum, which considered transaction amounts and timestamps and used an supported vector machine (SVM) [25] classifier to distinguish accounts. Xia et al. [16] proposed a novel node relabeling strategy based on Ethereum transaction attributes including transaction amount, number, and direction, and differentiating nodes and subgraphs by new labels. Yuan et al. [17] proposed an improved Graph2vec for phishing detection on Ethereum by learning the representation of transaction networks. These works learn some blockchain transaction network features, however, they do not take into consideration information related to Ethereum

transactions, such as transaction features and intensity of interaction between accounts.

C. Graph Neural Network

Chen et al. [13] designed phishing scams detection in ethereum transaction network (E-GCN) to detect phishing accounts, which was the first time introducing graph convolutional network (GCN) to phishing account detection on Ethereum. Li et al. [14] proposed a temporal transaction aggregation graph network (TTAGN) to enhance the detection performance of phishing accounts on Ethereum. Liu et al. [26] proposed a novel filter and augment graph neural network (FA-GNN) framework for classifying different accounts on Ethereum. Wang et al. [15] proposed a feature representation based heterogeneous network embedding method to identify Ethereum accounts. All of the above methods transform the Ethereum account classification problem into a node classification task and obtain satisfactory performance, but they have some shortcomings.

- 1) The input to the node classification task is the largest connected subgraph containing as many phishing accounts as possible, however, the Ethereum transaction network is large and sparse, so the constructed graph does not contain all phishing accounts.
- 2) When a new phishing account appears, it could not be included in the graph, so the added phishing account may not be detected.
- 3) Huge graphs lead to increased model training time, higher storage and processor requirements, and thus slower detection. In summary, some existing work suffers from high complexity, high latency, and poor scalability.

In this article, phishing detection is converted to a graph classification task, with the input being a local subgraph containing each target account to be detected and an augmented feature matrix. This has many advantages.

- 1) The training time is greatly decreased, while storage and processor requirements are reduced.
- 2) For an added phishing account, an ego-graph containing it can be constructed and fed into the model for detection, thus the scalability is enhanced.
- 3) Only the local subgraphs and augmented features of the account to be detected are acquired each time, so fast detection can be achieved.

III. METHOD

In this section, we first describe the problem to be addressed in this article and the feasibility of the proposed model. Then, we introduce the proposed model officially, and the framework is shown in Fig. 2. Our model consists of three modules: 1) data collection; 2) feature augmentation; and 3) graph representation learning.

In the data collection module, we first crawl the labeled phishing accounts from authoritative websites, then filter the normal accounts by some rules, and finally construct the ego-graph dataset consisting of phishing accounts and normal accounts. In the data augmentation module, we first compare and analyze the transaction behaviors of the two accounts, and then

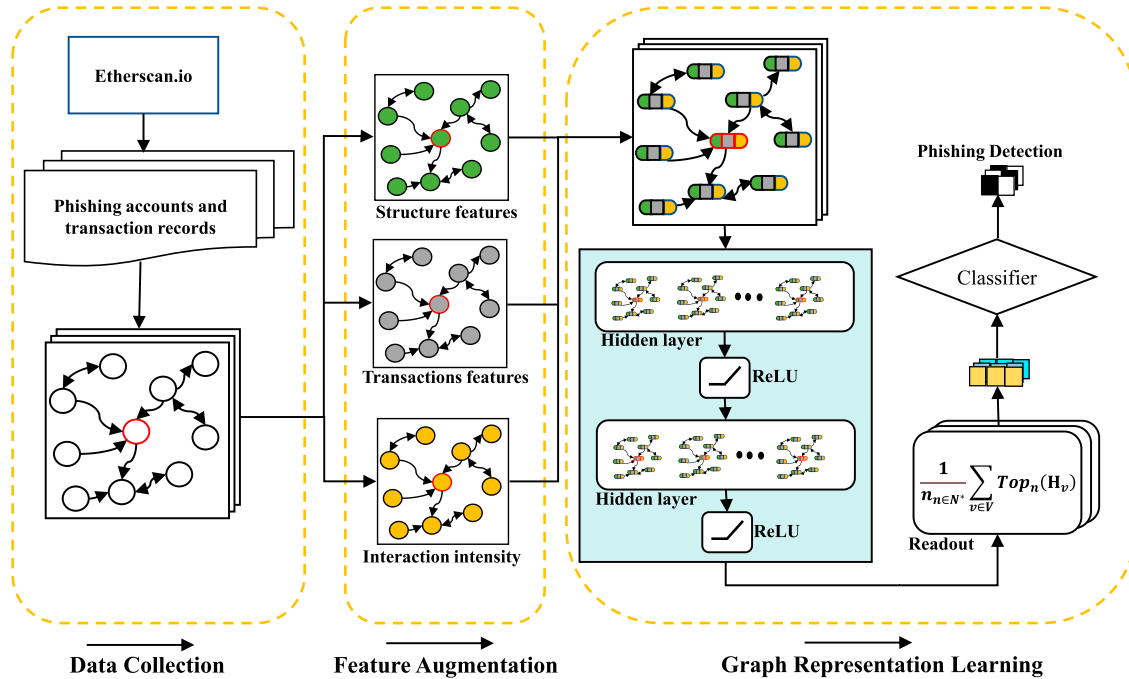


Fig. 2. Overall framework of PEAE-GNN.

design a feature augmentation strategy to enhance the node features, including structure features, transaction features and interaction intensity. In the graph representation learning module, first, we update each node representation with the graph neural message passing mechanism, and then a new readout operation is proposed to obtain the graph representation, which is finally used to classify the accounts.

A. Problem Statement

Although account-related user profiling information is unavailable due to the anonymity of the blockchain, we have summarized some differences between phishing accounts and normal accounts by analyzing the lifecycle of phishing activities on Ethereum.

First, from the directional topological graph structure of the transaction network, the transaction networks around the phishing addresses present like a converging star chart with the direction pointing to the phishing address from the outside. Second, from the interaction behavior, there is always only one transaction record between the phishing account and the victim's account, because the victim won't transfer Ether to the phishing account again after realizing being cheated. In addition, the transaction amounts of the phishing accounts involved are relatively large. At last, in terms of interaction intensity, the whole lifecycle of phishing accounts is short and frequent, and the number of in-transactions is much larger than out-transactions.

The transaction relationships between different accounts on Ethereum naturally form a directed topology network structure. In recent years, graph neural networks [27], [28], [29], [30], [31] develop rapidly and have attracted widespread attention due to their powerful modeling capabilities. They have been

successfully applied in various fields such as recommendation systems, community classification, protein classification, article classification, etc. Inspired by these research results, we intend to identify users by analyzing the transaction network they constitute. However, it would be impossible to achieve this by analyzing the entire Ethereum transaction network, which would be limited by storage. We aim to identify different accounts by combining the local topology and interaction patterns of the target accounts.

In this article, the Ethereum phishing scam detection is considered as a graph classification task. We define the ego-graph as $\mathbf{G} = (V, E)$, where V is the set of nodes, E is the set of edges, $v \in V$ represents an Ethereum account, $e_{uv} \in E$ represents an edge from node $u \in V$ to node $v \in V$, adjacency matrix $A \in \{0, 1\}^{|V| \times |V|}$, the node features are represented in a feature matrix $\mathbf{X} \in \mathbb{R}^{|V| \times d}$, where d is the feature dimension. We construct a set of K -hop ego-graphs $\mathbb{G} = \mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3, \dots, \mathbf{G}_m$ centered on each target address v_i . Specifically, an ego-graph is a K -order directed graph with the target address v_i as the center node and expands outward, so that the maximum distance between v_i and other nodes is K . Our goal is to learn a mapping function $\mathcal{F} : \mathbb{G} \rightarrow \mathbb{Y}$, which predicts the labels of the graph in \mathbb{G} . The set of labels \mathbb{Y} includes phishing addresses and normal addresses on Ethereum. The notations used in this article are listed in Table I.

B. Data Collection

Thanks to the openness of the Ethereum blockchain, we can use the client Geth,⁴ blockchain browser Etherscan, and other platforms to access all the transaction records, so that we can get

⁴<https://geth.ethereum.org/>

TABLE I
TERMS AND NOTATIONS

Symbol	Definition
\mathbf{G}, \mathbb{G}	Ego-graph, sets of ego-graphs
V	The set of nodes in a graph
u, v	Node $u, v \in V$
$\mathcal{N}(v)$	The neighbors of node v
E	The set of edges in a graph
e	An edge $e \in E$
A	The graph adjacency matrix
X	The feature matrix of a graph
x_i	The i th node feature vector, $x_i \in X$
d	The dimension of a node feature vector
\mathbb{Y}	The label of graph
$ \cdot $	The length of a set
M	The number of transactions
K	The order of network
$D_{\text{in}}, D_{\text{out}}$	In-degrees, out-degrees
\mathcal{AF}	A set of aggregation functions
$\text{MT}_{\text{in}}, \text{MT}_{\text{out}}$	The maximum amount ratio of in-transactions, out-transactions
$\text{RT}_{\text{in}}, \text{RT}_{\text{out}}$	The interaction intensity of in-transactions, out-transactions
$\sigma(\cdot)$	The ReLU activation function
l	The number of hidden layers
RF	Readout operation
Readout \mathbf{G}	Graph embedding representation

the Ethereum dataset of the model transaction network. However, the current transaction records of Ethereum are numerous, so we only focus on the labeled target addresses.

XBlock [32], a leading blockchain academic platform, made public in October 2021 a list of 1660 verified phishing addresses collected by 17 October 2019, with a large number of new phishing accounts added over time. Therefore, in this article, we obtain the latest phishing account labels and data from Etherscan. We first crawled the addresses that were labeled as phishing accounts, which were rigorously audited, and we got a total of 5430 labeled phishing addresses as of 13 July 2022. Then, we use the Etherscan application programming interface [33] to recursively execute breadth-first search (BFS) centered on these target addresses to obtain K – hop transaction records. Note that the specific transaction records include: the address that initiated and accepted the transaction, the transaction timestamp, and the transaction amount.

At present, some works study defense methods against adversarial samples [34], [35]. In this article, we randomly select normal addresses in the same way to construct the adversarial sample. Notably, numerous addresses on Ethereum are essentially worthless for the study target because of their extreme situation, and we refer to the data cleaning step of previous work [20] to eliminate some addresses with extreme situation to build a better model. Specifically, 1) We eliminated addresses with less than 5 and more than 2000 transaction records. On the one hand, a small number of records is not conducive to learning. On the other hand, addresses acting as

wallet interact with other addresses frequently, for which their numbers of neighbor addresses are too large for analysis of transaction network. This kind of data increases the burden of data processing, and even cause unexpected deviations to the results of model training [17]. 2) Filtering transaction records involving smart contract addresses. Smart contracts are often logically complex and do not facilitate phishing scams. 3) Filter out account addresses other than phishing accounts and normal accounts, such as mining accounts, InitialCoinOfferings (ICOs) and gambling accounts. These filtering rules allow the model to focus on learning the features of phishing scams.

C. Feature Augmentation

As shown in the second step in Fig. 2, we propose a feature augmentation component to generate augmented features for each node. Since there is no portrait information in the blockchain, in order to obtain higher-order information about the accounts, we analyze the transaction behavior of phishing accounts compared with normal accounts, and then propose a new feature augmentation strategy with three key components to augment the node features: 1) structure features; 2) transaction features; and 3) interaction intensity.

1) *Structure Features*: The structure features include the out-degree and in-degree of the node. Note that the out-degree refers to the number of times the current node sends Ether to other nodes, and the in-degree refers to the number of times other nodes send Ether to the current node. The structure features are defined in the following equation:

$$D_{\text{in}}(u) = \sum_{v \in V} |e_{vu}|, \quad D_{\text{out}}(u) = \sum_{v \in V} |e_{uv}|. \quad (1)$$

2) *Transaction Features*: Transaction features focus on the characteristics of transaction amounts between nodes, which in a way differentiate the nodes. Fig. 3 shows all the transactions of normal and phishing accounts throughout their lifecycle, respectively. The horizontal axis indicates time (unit: seconds), we normalize the lifecycle of the accounts to the same interval for comparison, and the size of the bubbles represents the size of the transaction amount. First, normal accounts show greater continuity and regularity, both in terms of in-transactions and out-transactions. This is easy to understand, as normal accounts have a more stable spending, while phishing accounts often lure others to transfer Ether with high return investments, which involve a much larger transaction amount. When the amount of Ether obtained reaches a certain size, the scammer transfers it in stages. We consider a set of aggregation functions, denoted by \mathcal{AF} , consisting of maximum, minimum, mean, median, standard deviation, and sum. Apply \mathcal{AF} to the in-transactions and out-transactions, denoted as \mathcal{AF}_{TO} and \mathcal{AF}_{TI} .

In addition, we also consider the ratio of maximum single transaction amount to the sum in different directions (i.e., out-transactions and in-transactions), defined as shown in the following equation:

$$\text{MT}_{\text{in}} = \frac{\text{Max}_{\text{in}}}{\text{Sum}_{\text{in}}}, \quad \text{MT}_{\text{out}} = \frac{\text{Max}_{\text{out}}}{\text{Sum}_{\text{out}}} \quad (2)$$

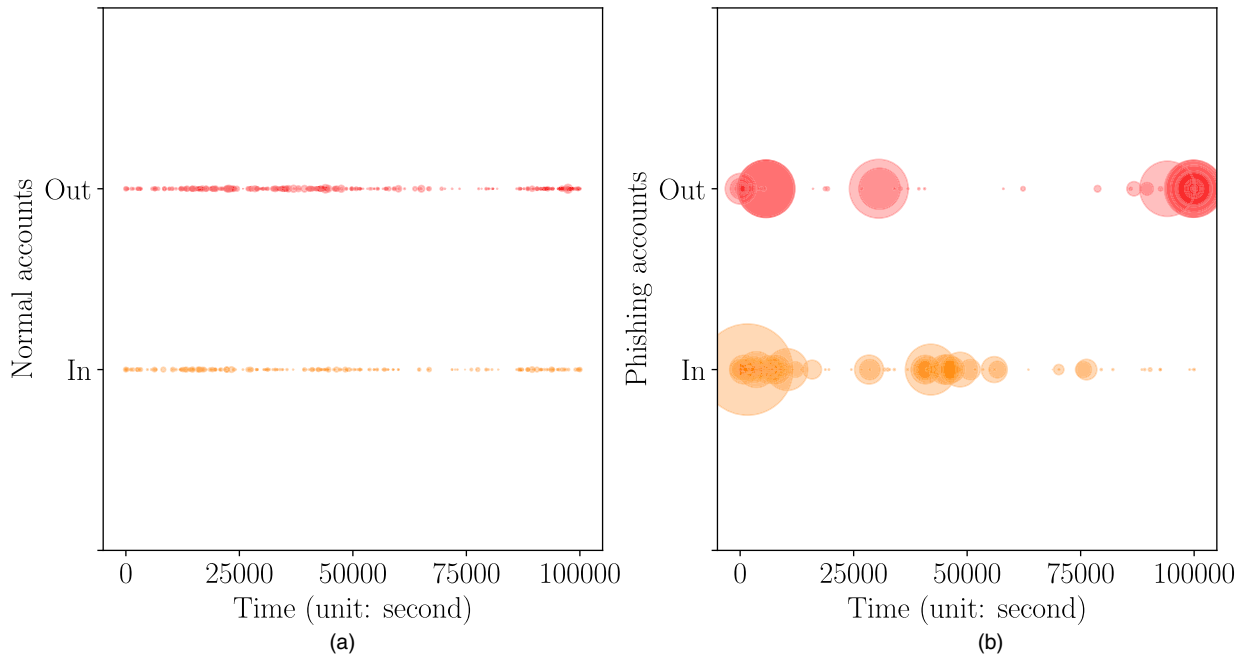


Fig. 3. (a) Normal accounts transactions and (b) phishing accounts transactions (red indicates out-transactions and orange indicates in-transactions).

3) *Interaction Intensity*: Different from structure and transaction features, interaction intensity considers the connection degree of nodes with their direct neighbors. We define the interaction intensity metric in the following equation in different directions, respectively:

$$\begin{aligned} RT_{in}(u) &= \frac{\sum_{\exists e_{vu} \in E, v \in V} |v|}{D_{in}(u)} \\ RT_{out}(u) &= \frac{\sum_{\exists e_{uv} \in E, v \in V} |v|}{D_{out}(u)}. \end{aligned} \quad (3)$$

Most of the transactions between phishing accounts and other accounts are one-time, i.e., one is usually scammed only once. Therefore, the phishing accounts do not have a fixed social circle, and the RT_{in} value of phishing accounts tends to 1. As shown in Fig. 4(a), compared with normal accounts, the phishing accounts showed a “top-heavy” phenomenon. The fraudsters must transfer digital currency out to his other normal accounts in stages, so the RT_{out} value of the phishing account also tends to 1, as shown in Fig. 4(b). In addition, we also use the aggregation function \mathcal{AF} for the transaction time interval between node u and its neighbor node v , denoted as \mathcal{AF}_{TS} .

Algorithm 1 describes the feature augmentation process with the entire ego-graph $\mathbf{G} = (V, E)$ and depth K as inputs. In the outer loop, K controls the depth in exploring the higher-order neighborhood. We consider two directions for all features except for the time interval. Inspired by the work of Liu et al. [36], to reduce the introduction of noise, we only keep structure features for nontarget nodes, see row 9. Finally, we obtain the enhanced feature matrix \mathbf{X} , and then use the \mathbf{X} as the input in the downstream task.

Algorithm 1 Feature augmentation.

Input: The ego-graph $\mathbf{G} = (V, E)$; depth K ;
Output: Enhanced features \mathbf{X}

- 1: $h_v^0 \leftarrow x_v, \forall v \in V$; // Initializing node features
- 2: for $k = 1, 2, \dots, K$ do
- 3: // Iterating over all nodes
- 4: for $v \in V$ do
- 5: if $v == \text{target node}$
- 6: // Concatenating features
- 7: $h_v^k \leftarrow \{D_{in}, D_{out}, \mathcal{AF}_{TO}, \mathcal{AF}_{TI}, MT_{in}, MT_{out}, RT_{in}, RT_{out}, \mathcal{AF}_{TS}\}$;
- 8: else
- 9: $h_v^k \leftarrow \{D_{in}, D_{out}\}$;
- 10: end if
- 11: end for
- 12: end for
- 13: $\mathbf{X} \leftarrow \text{Min-Max Normalization}(h_v^K), \forall v \in V$;
- 14: **return** \mathbf{X} ;

D. Graph Representation Learning

After the feature augmentation process, we get an enhanced feature set \mathbf{X} for each ego-graph $\mathbf{G} = (V, E)$. Then \mathbf{G} and \mathbf{X} are used as inputs to the neural network model, see the third step in Fig. 2.

The node information is transmitted to neighboring nodes and domain nodes through a message propagation mechanism, while mining the mutual information among nodes, i.e., the implicit information in the Ethereum transaction network. For a given \mathbf{G} , the neighboring nodes of each node are sampled and the feature set is aggregated by an aggregation function. In our model, the representation updating rule of the l th layer is shown

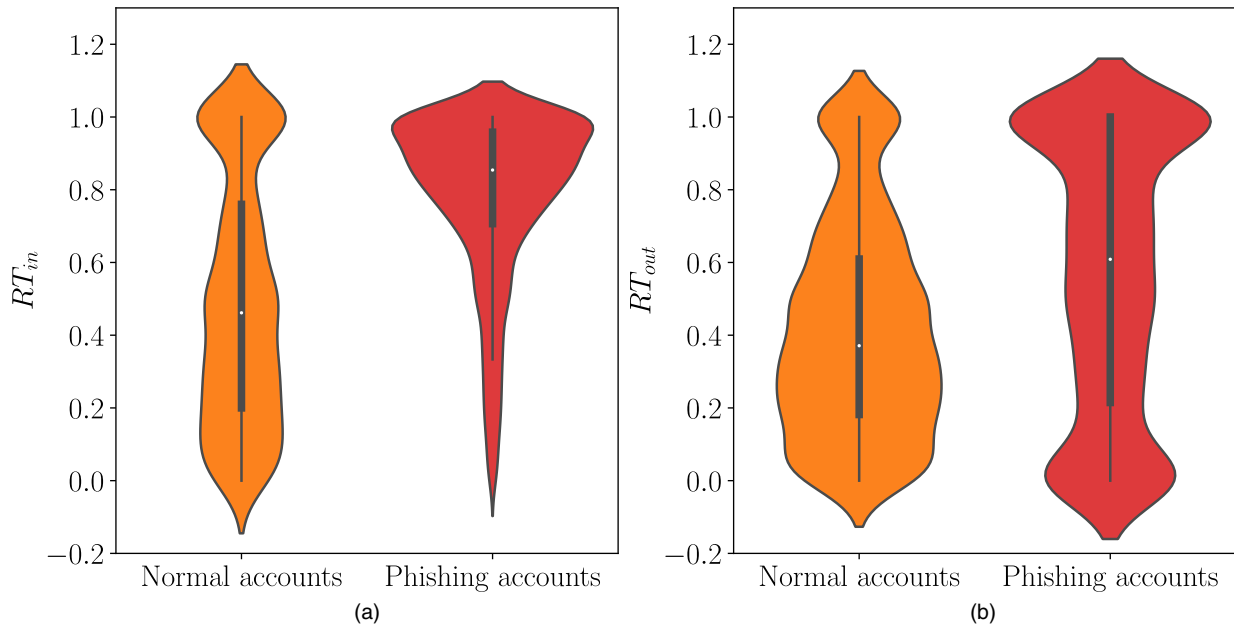


Fig. 4. Distribution of (a) RT_{in} and (b) RT_{out} values for normal and phishing accounts.

in the following equation:

$$\mathbf{H}_{\mathcal{N}(v)}^l = \sum_{u \in \mathcal{N}(v)} \frac{\mathbf{H}_u^{(l-1)}}{|\mathcal{N}(v)|}$$

$$\mathbf{H}_v^l = \sigma \left(\mathbf{W}^l \cdot \text{CONCAT} \left(\mathbf{H}_v^{l-1}, \mathbf{H}_{\mathcal{N}(v)}^l \right) + \mathbf{b}^l \right) \quad (4)$$

where $\mathcal{N}(v)$ denotes all neighbors of node v , \mathbf{H}_v^{l-1} denotes the embedding of node v in the previous layer, σ is a nonlinear activation function such as rectified linear unit (ReLU), \mathbf{W}^l is the weight matrix, and \mathbf{b}^l is the bias. CONCAT denotes the concatenation operation. The embedding of node v in layer l is obtained by aggregating the embeddings of all nodes u around node v . In addition, we add a self-loop to each node so that the node can aggregate its own information.

Our task is to predict the category of a graph. Therefore, a graph-level low-dimensional representation needs to be obtained after updating the nodes' representations. Readout is a step of extracting graph embedding considering all the node representations of the last updated graph. In general, graph embeddings can be obtained by simply summing or averaging all node representations. There is also a method of extracting the maximum value of each dimension, such as max pooling, which extracts the most important features from the convolutional neural network (CNN)-based model, as shown in

$$\text{Readout}_{\mathbf{G}} = \text{RF}(\mathbf{H}_v). \quad (5)$$

RF represents the summation, mean or maximum operation depending on the operation of readout function. In this article, we propose a new graph-level representation method, named RTM, which is an acronym for readout, top and mean, defined as shown in

$$\text{RTM} = \frac{1}{n_{n \in N^*}} \sum_{v \in V} \text{Top}(\mathbf{H}_v) \quad (6)$$

where \mathbf{H}_v is the representation of all nodes in the graph $\mathbf{G} \in \mathbb{G}$ after the neural network, and Top_n is the top n nodes after the descending sorting. Specifically, we sort the final obtained node representations in descending order, sum the top n values, and compute the mean value as the final graph \mathbf{G} representation $\text{Readout}_{\mathbf{G}}$.

To train our model, we used the cross-entropy loss function as the objective function for optimization, as shown in the following equation:

$$\mathcal{L} = -\frac{1}{n} \sum_{i=1}^N (y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i)) \quad (7)$$

where \hat{y}_i represents the prediction of the i th graph and y_i represents the ground truth.

E. Model Analysis

1) *Time Complexity Analysis*: The time cost of PEA-E-GNN mainly comes from the feature augmentation and feature aggregation, and the computational complexity in feature augmentation is $O(|V|^K)$, $K \in \mathbb{N}^*$, where K , $|V|$ denote the depth and the total number of nodes. The time complexity in feature aggregation is $O(|V|d^2 \prod_{i=1}^l S_i)$, $i \in \{1, 2, \dots, l\}$, where S_i is the number of aggregated neighbors at the i th layer, d is the dimensions of the node hidden features remain constant. In total, the time complexity of the whole training cycle is $O(|V|^K + |V|d^2 \prod_{i=1}^l S_i)$. In this article, $K = 2 = 2$ and $l = 2$, so the time complexity of PEA-E-GNN is $O(|V|^2 + |V|d^2 S_1 S_2)$.

2) *Memory Complexity Analysis*: Some studies have shown that the full-batch training algorithm suffers significantly from memory overflow problem [37]. For example, GCN [27] is usually required to save the whole graph data and intermediate states of all nodes into memory. GCN memory complexity is $O(l|V|d + ld^2)$. The memory complexity of PEA-E-GNN is

TABLE II
STATISTICS OF THE DATASETS

Datasets	Classification	G Number	Properties	Node Number	Edge Number	Degrees
EthereumD1	Phishing	2348	Average	790.28	3947.07	4.28
			Maximum	28447	166623	46
	Normal	4475	Average	887.8	3099.7	6.27
			Maximum	34334	139063	226
EthereumD2	Phishing	801	Average	817.51	2767.48	3.53
			Maximum	28437	138474	53
	Normal	807	Average	2251.58	7945.57	4.95
			Maximum	54025	529788	162

$O(sd\prod_{i=1}^l S_i + ld^2)$, $i \in \{1, 2, \dots, l\}$, where s is the batch size and l is the number of layers.

IV. EXPERIMENTS

In this section, we show the experimental results of the proposed framework. First, we introduce the datasets used in this article. Second, we explain the parameter settings and the experimental environment in detail. Then, to demonstrate the effectiveness, parameter sensitivity, scalability of the proposed algorithm, we perform a phishing detection on the Ethereum transaction data and analyze the experimental results. Next, we conduct ablation experiments to explore the importance of the three feature enhancement components. Finally, we perform a case study to demonstrate the unique advantages of PEAE-GNN compared to other methods.

A. Datasets

After filtering, we construct the largest Ethereum phishing detection dataset yet, named EthereumD1, which contains a total of 2348 phishing accounts and 4475 normal accounts, which are then used as input, and each graph corresponds to the label of its central node. Our dataset is available at <https://github.com/King-Roland/PEAE-GNN>. In addition, we use the second-order phishing node transaction network dataset provided by the blockchain academic platform XBlock [32], named EthereumD2. We conduct experiments using EthereumD1 and EthereumD2 to fully evaluate the performance of our proposed PEAE-GNN model, and detailed data information is provided in Table II.

B. Baselines and Experimental Setup

To evaluate the performance of the proposed PEAE-GNN for detecting phishing accounts, we compare eleven representative methods as follows. In particular, logistic regression (LR) and SVM are traditional machine learning methods. Node2vec and Trans2vec are the node embedding model. SF, Graph2vec, and Line_Graph2Vec are the graph embedding model. I2BGNN is the work of Shen et al. [18] and the last three are graph neural network models. The detailed descriptions are as follows.

1) *LR* [38]: LR is a simple and powerful approach to solve some linear binary classification problems. It is used in the detection of phishing scam [24].

- 2) *SVM*: SVM is a powerful machine learning approach used in phishing scam accounts detection on blockchain [20], [39].
- 3) *Node2vec* [40]: Node2vec is similar to DeepWalk [41], which maintains high-order proximity between nodes by maximizing the probability of occurrence of subsequent nodes in a fixed-length random walk. The main difference from DeepWalk is that node2vec uses biased random wandering and trades off between breadth-first (BFS) and depth-first (DFS) graph search, so Node2vec produces embeddings with higher quality and more information than DeepWalk. Choosing the right tradeoff allows Node2vec to maintain community structure as well as structural equivalence between nodes.
- 4) *Trans2vec* [24]: Trans2vec is an improved variant of Node2vec, which is based on transaction amount and timestamp to calculate transition probabilities.
- 5) *SF* [42]: SF authors define the normalized Laplace operator of the graph to extract the graph features, which relies on the graph's spectral features.
- 6) *Graph2vec* [43]: Graph2vec transfers the document embedding approach to graph classification and learns a distributed representation of the entire graph by a document embedding neural network. Graph2Vec first extracts the rooted subgraphs and saves the corresponding labels into a vocabulary, then trains a skip-gram model to obtain a representation of the entire network. The generated embeddings are learned in an unsupervised manner.
- 7) *Line_Graph2Vec* [17]: Line_Graph2Vec is a method based on graph2vec graph embedding and line graph. The graph2vec treats the rooted subgraphs around the nodes as words and uses the skip-Gram model to generate the embedding vectors.
- 8) *I2BGNN* [18]: I2BGNN analyzes users' behavior from the perspective of transaction subgraph, extracting transactional subgraphs for each labeled account by a sampling mechanism. This end-to-end model is proposed to implement identity inference on the blockchain using graph neural networks.
- 9) *GCN* [27]: GCN is the first graph convolutional network implemented by a local first-order approximation of spectral graph convolution. The representation of the hidden layer is performed by encoding the local graph structure as well as the node features.
- 10) *Graph attention network (GAT)* [28]: GAT adopts a self-attentive mechanism to specify the weights of neighboring nodes and to aggregate neighbors. It is popular in many tasks.
- 11) *Graph isomorphism network (GIN)* [30]: GIN is a powerful GNN in the neighborhood aggregation framework, which only uses MLP and summation. Its performance in distinguishing graph structures is comparable to WL-Test.

Some related works could not be compared because their source codes and data are not available. we will disclose all related codes and data after the article is published.

1) *Method Setting*: In our method, $K = 2$ and $l = 2$, i.e., extracting 2-hop ego-graph and using 2 hidden layers, we use

TABLE III
PERFORMANCE COMPARISON OF DIFFERENT METHODS

Method	Readout	EthereumD1			EthereumD2		
		Precision	Recall	F1-Score	Precision	Recall	F1-Score
Line_Graph2Vec [17]	–	–	–	–	0.69	0.77	0.73
Node2vec [40]	–	0.6949	0.8686	0.7721	0.6363	0.6176	0.6268
SVM [20]	–	0.7212	0.7661	0.743	0.7748	0.7653	0.77
LR [38]	–	0.7896	0.8195	0.8043	0.7962	0.8289	0.8122
SF [42]	–	0.8053	0.7778	0.7913	0.6806	0.5833	0.6282
Graph2vec [43]	–	0.8403	0.8547	0.8475	0.8344	0.8844	0.8587
Trans2vec [24]	–	0.8175	0.8876	0.8512	0.7865	0.8131	0.7996
I2BGNN [18]	–	0.8662	0.8967	0.8812	0.8571	0.8835	0.8701
GCN [27]	Max	0.833	0.8659	0.8491	0.6771	0.7831	0.7263
	Sum	0.8188	0.7846	0.8013	0.694	0.7651	0.7278
	Mean	0.7783	0.8022	0.79	0.6667	0.756	0.7085
	RTM	0.8783	0.8769	0.8778	0.6813	0.747	0.7126
GAT [28]	Max	0.8786	0.8747	0.8767	0.7662	0.7108	0.7375
	Sum	0.8868	0.74066	0.8072	0.6976	0.8614	0.7709
	Mean	0.7995	0.7011	0.7471	0.7256	0.7169	0.7212
	RTM	0.8969	0.9186	0.9077	0.7919	0.7108	0.7492
GIN [30]	Max	0.8881	0.7846	0.8331	0.7985	0.6446	0.7133
	Sum	0.8946	0.8022	0.8459	0.7667	0.6928	0.7278
	Mean	0.8435	0.8527	0.8481	0.7452	0.7048	0.7245
	RTM	0.9159	0.9098	0.9128	0.8095	0.7169	0.7604
PEAE-GNN (Ours)	Max	<u>0.919</u>	0.9231	<u>0.9211</u>	0.9487	0.8916	0.9193
	Sum	0.913	0.8769	0.8946	0.9006	0.939	<u>0.9194</u>
	Mean	0.8977	<u>0.9253</u>	0.9113	0.8654	0.8232	0.8437
	RTM	0.9336	0.9274	0.9305	<u>0.9277</u>	0.939	0.9333

Bold indicates the first best result and underline indicates the second best result.

Adam [44] as the optimization method and the learning rate is set to 0.001. we randomly divide the data into train and test sets by ratio of 8:2. And we uniformly adopt the under sampling solution for all methods. For LR and SVM, the max_iter is set to 300. For Node2vec, Graph2vec, and SF, the embedding dimension is set to 128, random seed is set to 42, epoch is set to 64, and other parameters use the default settings. For PEAE-GNN, batch size is set to 32, epoch is set to 64, hidden layer dimension is 128, and the parameter settings of other graph neural network models keep the same as ours. For GAT, we set Attention heads to be 3. For GIN, we use a single layer perceptron with mean aggregator. In our experiments, we set the parameters of I2BGNN as suggested in the original article: 2 hidden layers, output dimension set to 128, and set the maximum epoch number to 50, batch size to 30, and dropout to 0.3. Finally, all methods use the same classifier.

2) *Experimental Environment*: We use PyTorch [45] to implement our proposed PEAE-GNN, and our experimental environment consists of Intel Core i9-10940X CPU @ 3.30GHz, NVIDIA GeForce RTX 2080 Ti × 2, 16GBx4 memory (DDR4) and Windows 10 Professional (OS).

C. Evaluation Metrics

In this article, we provide a comprehensive evaluation on the performance of different methods using the following three metrics, and the prediction results can be divided into the following four situations based on the true and prediction labels of the ego-graph.

- 1) *True positive (TP)*: The true labels are phishing accounts, and the predicted labels are phishing accounts.

- 2) *False positive (FP)*: The true labels are normal accounts, and the predicted labels are phishing accounts.
- 3) *False negative (FN)*: The true labels are phishing accounts, and the predicted labels are normal accounts.
- 4) *True negative (TN)*: The true labels are normal accounts, and the predicted labels are normal accounts.

Precision: It shows how many of the samples with positive category are real positive samples and is defined as

$$\text{Precision} = \frac{TP}{TP + FP}.$$

Recall: Recall indicates how many positive examples in the sample are predicted correctly and is defined as

$$\text{Recall} = \frac{TP}{TP + FN}.$$

F1-score: The F1-score is the harmonic average of precision and recall without being influenced by unbalanced samples. The calculation formula is

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

D. Results and Analysis

1) *Classification Performance*: In this subsection, we present a comprehensive evaluation and analysis of proposed model. Table III summarizes the experimental results of our model on two datasets. The best results are depicted in bold, and the suboptimal results are underlined.

Among all the baselines, using Node2vec to obtain node embeddings exhibits the worst performance due to the fact

that Node2vec is a representation learning method based on low-dimensional structures. It uses the similarity information of neighboring nodes to train the representation vector, by which only local information can be extracted. Compared with Node2vec, SF, and Graph2vec can learn the structure information of local and global similarity between graphs, so it has some performance improvement for phishing detection. However, none of the three methods above takes full advantage of the rich transaction information between accounts, so there is room for performance improvement. GCN aggregates neighbor information during the learning process, which propagates and amplifies errors as the neural network goes deeper. Although GAT adopts an attention mechanism for neighbor aggregation, the attention scores do not work well in the Ethereum phishing detection task, where the neighbors of phishing accounts are all normal accounts. We note that GIN outperforms the GAT method slightly, but may introduce noise and compress useful information when facing complex Ethereum transaction networks. It is worth noting that readout using the mean operation performs the worst for Ethernet phishing detection. In addition, most phishing account features have larger values than normal accounts, with some exceptions where normal accounts have a high volume of transactions. the RTM readout operation is a compromise between others, and it can retain valid information and avoid the introduction of noise. We note that using max and sum readout performs better on EthereumD2, and the possible reason is due to the selection of normal user samples in EthereumD2.

In addition, we study the performance of PEAE-GNN in different stages of the phishing account lifecycle, and we set a parameter β , $\beta \in [0.6, 0.7, 0.8, 0.9]$. For an Ethereum account u with M transactions, the lifecycle of u is $\text{Life}_u = t_M - t_1$, where t_M denotes the time for the last transaction, t_1 denotes the time for the first transaction.

We conduct experiments using different stages of the phishing account lifecycle, specifically, constructing a transaction network graph using phishing account $\beta \times \text{Life}_u$ and then feeding it into our model, and the results are shown in Fig. 5, our method still shows good performance despite the β of 0.6. As the β value becomes larger, the difference in trading network topology and trading behavior patterns between the phishing account and the normal account becomes larger. On the EthereumD1 and EthereumD2 datasets, compared with β taken as 0.6, our model improves the accuracy by 5.46% and 9.09%, recall by 3.58% and 9.35%, and F1-score by 5.06% and 9.28%, respectively, when β taken as 0.8. The experimental results show that our method can detect phishing accounts earlier and thus prevent more people from being scammed. Our method can be added to the browser as a plugin which allows phishing detection of the recipient when the user transfers Ether.

2) *Parameter Sensitivity*: We conduct extensive experiments to investigate the parametric sensitivity of some important parameters in our model.

As shown in Fig. 6, the best results are obtained when the number of hidden layers is 2, and the results deteriorate as the number of hidden layers increases. The deterioration may be due to the fact that the graph convolution contains the operation

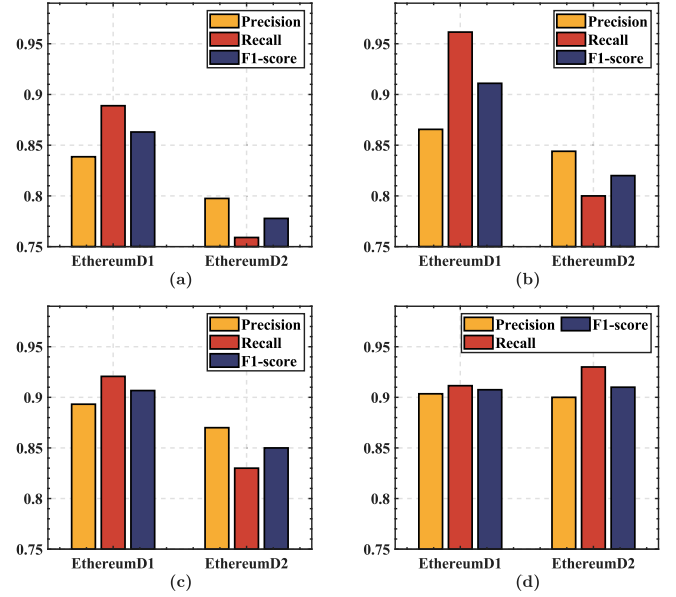


Fig. 5. Experimental results with different values of β on EthereumD1 and EthereumD2 datasets. (a) $\beta = 0.6$. (b) $\beta = 0.7$. (c) $\beta = 0.8$. (d) $\beta = 0.9$.

of aggregating the neighboring nodes' features, and the features between nodes become too smooth with too many layers, leading to a lack of differentiation and also amplifying the effect of noise. Fig. 7 shows the effect of different neuron number in hidden layer on Ethereum phishing detection, and we can see from the plot that the model performs best when the hidden layer dimension is 128. We notice that the training loss do not decrease when using a small number of neurons and the model underfits. As the number of neurons increases, the model performs better on the training set but worse on the test set, and the model overfits. In addition, the training time of the model increases significantly and the memory usage is larger.

To make a comprehensive evaluation, we use different partitioning of the datasets. Experiments with different partitions of the dataset illustrate that our method can still perform well with a small training set. Specifically, we split it in four ways, the ratio of training set and test set are 3:7, 5:5, 7:3, and 8:2 respectively, which are then fed into the model to test the performance. As shown in Fig. 8, our model still has a good performance despite in 3:7, achieving an F1-score of 0.83, which indicates that our model has a good learning ability.

Next, for our proposed graph representation, different values of n have an impact on the final result, so we investigate the performance with different settings of n . Specifically, $n \in [1, 2, 3, 4, 5]$. As shown in Fig. 9, the model shows the best results when $n = 3$. For the Ethereum phishing detection, we find that the enhanced feature values of phishing accounts are larger than those of normal accounts overall. However, if we just use the maximum readout there may be insufficient representation power to the point of weak differentiation. So, we heuristically sort the aggregated node features in descending order and then take the mean value of the top n to get the graph representation. The experimental results show that too large or too small a value of n degrades the performance, the reason

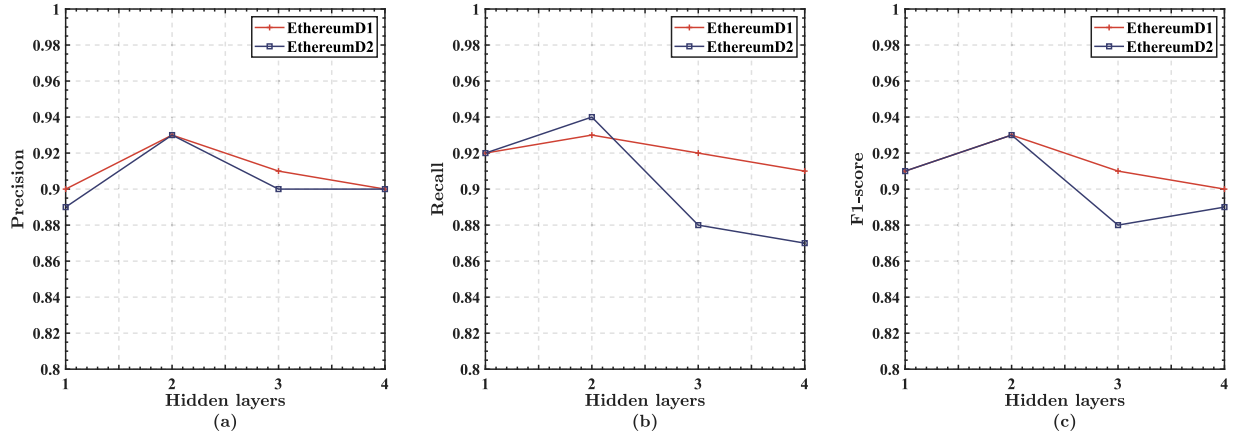


Fig. 6. Experimental results with different number of hidden layers on EthereumD1 and EthereumD2 datasets. (a) Precision. (b) Recall. (c) F1-score.

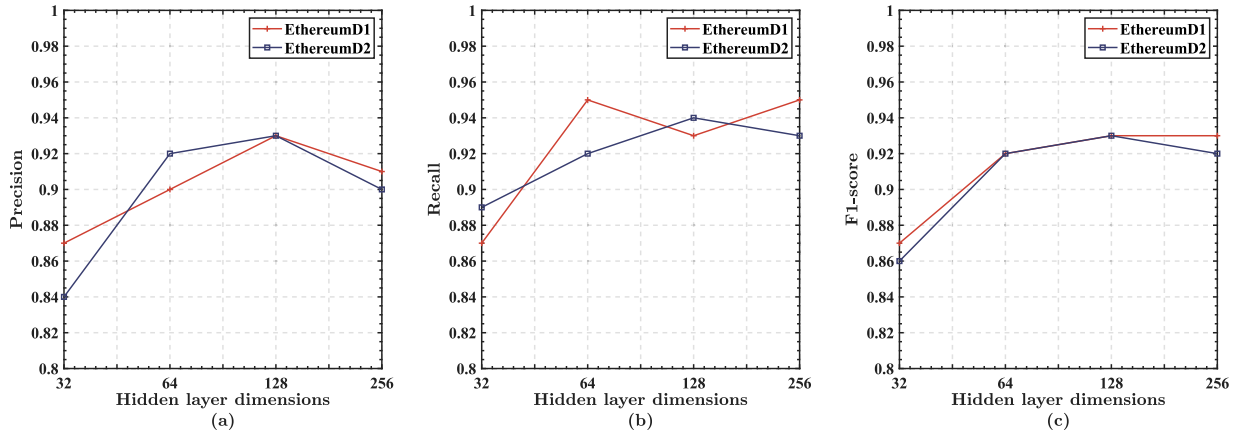


Fig. 7. Experimental results with different dimensions of the hidden layer on EthereumD1 and EthereumD2 datasets. (a) Precision. (b) Recall. (c) F1-score.

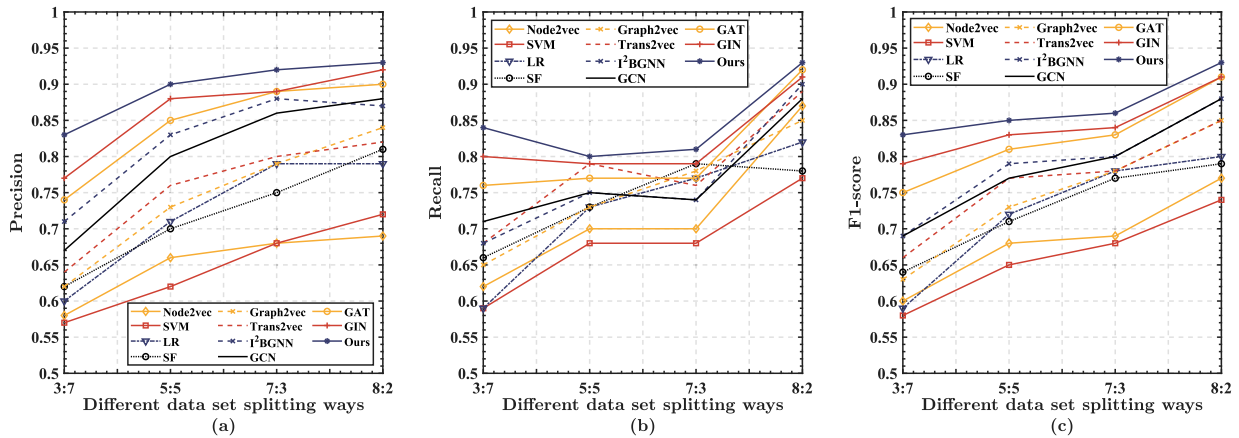


Fig. 8. Experimental results with different dataset splitting ways on EthereumD1. (a) Precision. (b) Recall. (c) F1-score.

may be that if the value of n is too small, the representation capability of the obtained graph representation is insufficient, and if the value of n is too large, noise will be introduced.

3) *Ablation Experiments*: To explore the contribution of the three feature augmentation modules, we conduct ablation

experiments for our model. Table IV shows the corresponding contribution of each module. Through comparison of the different modules, we can observe that only retaining transaction features has the greatest improvement on the different metrics, the reason for this is that the pattern of trading behavior of

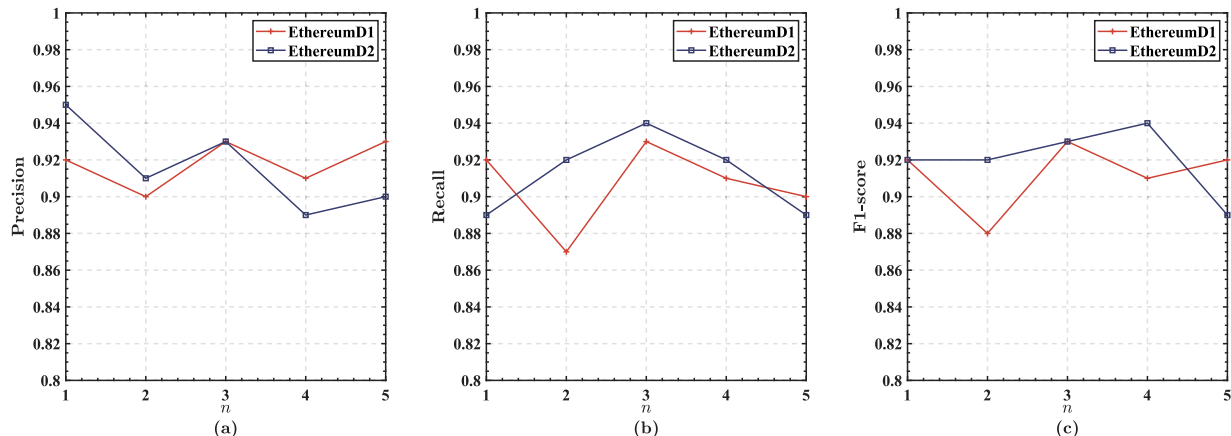
Fig. 9. Experimental results with different n values on EthereumD1 and EthereumD2 datasets. (a) Precision. (b) Recall. (c) F-score.

TABLE IV
PERFORMANCE COMPARISON OF DIFFERENT FEATURE
AUGMENTATION MODULES

Perspective Detail	EthereumD1			EthereumD2		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Structure features Only	0.7744 ↓ (17.1%)	0.7846 ↓ (15.4%)	0.7794 ↓ (16.2%)	0.7416 ↓ (20.1%)	0.8049 ↓ (14.3%)	0.7719 ↓ (17.3%)
Transactions features Only	0.84 ↓ (10%)	0.8659 ↓ (6.6%)	0.8528 ↓ (8.4%)	0.8039 ↓ (13.3%)	0.741 ↓ (21.1%)	0.7712 ↓ (17.4%)
Interaction intensity Only	0.8308 ↓ (11%)	0.8637 ↓ (6.9%)	0.8469 ↓ (9%)	0.7682 ↓ (17.2%)	0.6988 ↓ (25.6%)	0.7319 ↓ (21.6%)
Structure features and transactions features	0.9083 ↓ (2.7%)	0.9143 ↓ (1.4%)	0.9113 ↓ (2.1%)	0.8526 ↓ (8.1%)	0.8264 ↓ (12%)	0.8393 ↓ (10.1%)
Structure features and interaction intensity	0.8911 ↓ (4.6%)	0.8989 ↓ (3.1%)	0.895 ↓ (3.8%)	0.8634 ↓ (6.9%)	0.9133 ↓ (2.7%)	0.8877 ↓ (4.9%)
Transactions features and interaction intensity	0.9136 ↓ (2.1%)	0.9297 ↑ (0.3%)	0.9216 ↓ (1%)	0.8817 ↓ (5%)	0.9085 ↓ (3.3%)	0.8949 ↓ (4.1%)
Full	0.9336	0.9274	0.9305	0.9277	0.939	0.9333

phishing accounts is more different from that of normal accounts. The remaining modules improve each metric in different degrees, where only retaining the structure features shows the smallest improvement, which is easy to understand because it cannot be excluded that our normal samples include accounts like exchanges, which are frequently traded and therefore have large structure feature values, thus affecting the model's ability to distinguish phishing accounts from normal accounts. Our method still has some detection power for phishing accounts only considering structure features, since the model learns some structure features of the graph. We conduct extensive experiments on each ablation method to optimize all metrics as much as possible, and interestingly when only the structure features are removed, the recall metric has less than 0.3% improvement on the EthereumD1 dataset, and the accuracy and F1 values decrease by about 2.1% and 1%, respectively. In summary, the transaction feature is the most important in phishing account detection, followed by the interaction intensity and the structure feature last.

4) *Case Study*: We conduct a case study to demonstrate the unique advantages of PEAE-GNN compared to other methods. Table V shows a typical example of phishing detection, with a prediction result of 1 for normal accounts and -1 for phishing accounts. For phishing accounts with addresses

TABLE V
TYPICAL EXAMPLE OF PHISHING DETECTION

Method	Input	Prediction Results
SVM	Feature matrix X	1
LR	Feature matrix X	1
Node2vec	Directed ego-graph	1
Trans2Vec	Directed ego-graph	1
SF	Undirected ego-graph	1
Graph2vec	Undirected ego-graph	1
I2BGNN	Undirected ego-graph, feature matrix X	1
GCN	Directed ego-graph, feature matrix X	1
GAT	Directed ego-graph, feature matrix X	1
GIN	Directed ego-graph, feature matrix X	1
Ours	Directed ego-graph, enhanced feature matrix X	-1

“0xf9c8c57fe9eaabc5dfef8a8d03ef308f2a28c91c,” “0xdf9191889649c442836ef55de5036a7b694115b6,” and “0xa475a250519fd5d13682d1a380ce2dcabb8abc” etc., only PEAE-GNN can successfully detect them. We analyze these accounts and find that their number of transactions and transaction amounts are similar to normal accounts, which demonstrate the advantages of the PEAE-GNN feature augmentation module. For SVM and LR, the input data is the feature matrix. For Trans2vec, we follow the original article and perform biased sampling to obtain the node embedding. For the graph embedding model SF, Graph2vec, we convert the ego-graph to an undirected graph as the input. For I2BGNN, we follow the original proposal using an undirected graph, i.e., converting the directed adjacency matrix into a symmetric adjacency matrix as input.

V. CONCLUSION

In this article, we propose an Ethereum phishing detection model called PEAE-GNN. By analyzing the transaction characteristics of Ethereum phishing accounts, we find some differences in the transaction behavior of phishing accounts and normal accounts on Ethereum. Specifically, we propose a node feature augmentation strategy based on structure features, transaction features, and interaction intensity. Extensive experimental results demonstrate the superiority of our model for phishing detection, and that our model can detect phishing

accounts at an early stage. A possible application scenario is to embed our approach in a browser as a plugin to provide real-time monitoring and risk alerting for user transfers.

In the future, we will further investigate the detection of other identity accounts in blockchain, such as gambling accounts, mining accounts, etc. In addition, the detection of other accounts owned by fraudsters is also a research direction. Finally, to accelerate the research in this field, all relevant data and code will be open-sourced after the publication of the article.

REFERENCES

- [1] M. Khonji, Y. Iraqi, and A. Jones, "Phishing detection: A literature survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2091–2121, Fourth Quarter 2013.
- [2] E. Zhu, Z. Chen, J. Cui, and H. Zhong, "MOE/RF: A novel phishing detection model based on revised multi-objective evolution optimization algorithm and random forest," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 4, pp. 4461–4478, Dec. 2022.
- [3] L. Wang, H. Cheng, Z. Zheng, A. Yang, and X. Zhu, "Ponzi scheme detection via oversampling-based long short-term memory for smart contracts," *Knowl.-Based Syst.*, vol. 228, Sep. 2021, Art. no. 107312.
- [4] Y. Zhang, W. Yu, Z. Li, S. Raza, and H. Cao, "Detecting ethereum Ponzi schemes based on improved lightGBM algorithm," *IEEE Trans. Comput. Social Syst.*, vol. 9, no. 2, pp. 624–637, Apr. 2022.
- [5] "The chainalysis 2021 crypto crime report," Chainalysis, New York, NY, USA, 2021. [Online]. Available: <https://go.chainalysis.com/2021-Crypto-Crime-Report.html>
- [6] H. Chen, M. Pendleton, L. Njilla, and S. Xu, "A survey on ethereum systems security: Vulnerabilities, attacks, and defenses," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 1–43, 2020.
- [7] A. Trozze et al., "Cryptocurrencies and future financial crime," *Crime Sci.*, vol. 11, no. 1, pp. 1–35, 2022.
- [8] N. Anita and M. Vijayalakshmi, "Blockchain security attack: A brief survey," in *Proc. IEEE 10th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, 2019, pp. 1–6.
- [9] B. Huang, J. Liu, J. Wu, Q. Li, and H. Lin, "Temporal analysis of transaction ego networks with different labels on ethereum," 2022, *arXiv:2204.13253*.
- [10] D. Chaudhari, R. Agarwal, and S. K. Shukla, "Towards malicious address identification in bitcoin," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, 2021, pp. 425–432.
- [11] Y.-J. Lin, P.-W. Wu, C.-H. Hsu, I.-P. Tu, and S.-w. Liao, "An evaluation of bitcoin address classification based on transaction history summarization," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, 2019, pp. 302–310.
- [12] H. Wen, J. Fang, J. Wu, and Z. Zheng, "Hide and seek: An adversarial hiding approach against phishing detection on ethereum," *IEEE Trans. Comput. Social Syst.*, vol. 10, no. 6, pp. 3512–3523, Dec. 2023.
- [13] L. Chen, J. Peng, Y. Liu, J. Li, F. Xie, and Z. Zheng, "Phishing scams detection in ethereum transaction network," *ACM Trans. Internet Technol.*, vol. 21, no. 1, pp. 1–16, 2020.
- [14] 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 *Proc. ACM Web Conf.*, 2022, pp. 661–669.
- [15] Y. Wang, Z. Liu, J. Xu, and W. Yan, "Heterogeneous network representation learning approach for ethereum identity identification," *IEEE Trans. Comput. Social Syst.*, vol. 10, no. 3, pp. 890–899, Jun. 2023.
- [16] Y. Xia, J. Liu, and J. Wu, "Phishing detection on ethereum via attributed ego-graph embedding," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 5, pp. 2538–2542, May 2022.
- [17] Z. Yuan, Q. Yuan, and J. Wu, "Phishing detection on ethereum via learning representation of transaction subgraphs," in *Proc. Int. Conf. Blockchain Trustworthy Syst.*, Springer-Verlag, 2020, pp. 178–191.
- [18] J. Shen, J. Zhou, Y. Xie, S. Yu, and Q. Xuan, "Identity inference on blockchain using graph neural network," in *Proc. Int. Conf. Blockchain Trustworthy Syst.*, Springer-Verlag, pp. 3–17.
- [19] D. Zhang, J. Chen, and X. Lu, "Blockchain phishing scam detection via multi-channel graph classification," in *Proc. Int. Conf. Blockchain Trustworthy Syst.*, Springer-Verlag, 2021, pp. 241–256.
- [20] W. Chen, X. Guo, Z. Chen, Z. Zheng, and Y. Lu, "Phishing scam detection on ethereum: Towards financial security for blockchain ecosystem," in *Proc. Int. Joint Conf. Artif. Intell.*, 2020, pp. 4506–4512.
- [21] R. F. Ibrahim, A. M. Elian, and M. Ababneh, "Illicit account detection in the ethereum blockchain using machine learning," in *Proc. IEEE Int. Conf. Inf. Technol. (ICIT)*, 2021, pp. 488–493.
- [22] M. Weber et al., "Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics," 2019, *arXiv:1908.02591*.
- [23] M. A. Prado-Romero, C. Doerr, and A. Gago-Alonso, "Discovering bitcoin mixing using anomaly detection," in *Proc. Iberoamerican Congr. Pattern Recognit. Conf.*, Springer-Verlag, 2018, pp. 534–541.
- [24] J. Wu et al., "Who are the phishers? Phishing scam detection on ethereum via network embedding," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 2, pp. 1156–1166, Feb. 2020.
- [25] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intell. Syst. Appl.*, vol. 13, no. 4, pp. 18–28, Jul.–Aug. 1998.
- [26] J. Liu, J. Zheng, J. Wu, and Z. Zheng, "FA-GNN: Filter and augment graph neural networks for account classification in ethereum," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 4, pp. 2579–2588, Jul.–Aug. 2022.
- [27] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.
- [28] P. Velicković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*.
- [29] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30.
- [30] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" 2018, *arXiv:1810.00826*.
- [31] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.
- [32] "XBLOCK." Accessed: Jan. 2024. [Online]. Available: <http://xblock.pro/#/>
- [33] "Etherscan." Accessed: Jan. 2024. [Online]. Available: <https://docs.etherscan.io/>
- [34] J. Chen, H. Zheng, W. Shangguan, L. Liu, and S. Ji, "Act-detector: Adaptive channel transformation-based light-weighted detector for adversarial attacks," *Inf. Sci.*, vol. 564, pp. 163–192, Jul. 2021.
- [35] H. Zheng, J. Chen, H. Du, W. Zhu, S. Ji, and X. Zhang, "GRIP-GAN: An attack-free defense through general robust inverse perturbation," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 6, pp. 4204–4224, Nov.–Dec. 2022.
- [36] Z. Liu et al., "Poster: Neural network-based graph embedding for malicious accounts detection," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 2543–2545.
- [37] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.
- [38] D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein, and M. Klein, *Logistic Regression*. New York, NY, USA: Springer-Verlag, 2002.
- [39] H. Wen, J. Fang, J. Wu, and Z. Zheng, "Transaction-based hidden strategies against general phishing detection framework on ethereum," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2021, pp. 1–5.
- [40] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 855–864.
- [41] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 701–710.
- [42] N. de Lara and E. Pineau, "A simple baseline algorithm for graph classification," 2018, *arXiv:1810.09155*.
- [43] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "Graph2vec: Learning distributed representations of graphs," 2017, *arXiv:1707.05005*.
- [44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
- [45] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, vol. 32.



Hexiang Huang received the B.E. degree in software engineering from Hubei University of Automotive Technology, Shiyan, China, in 2020. He is currently working toward the M.E. degree with the School of Software, Yunnan University, Kunming, China.

His current research interests include blockchain security and deep learning.



Xuan Zhang (Member, IEEE) received the B.S. and M.S. degrees in computer science and the Ph.D. degree in system analysis and integration from Yunnan University, Kunming, China.

She is a Professor with the School of Software, Yunnan University, Yunnan, China. She is an author of three books and more than 100 articles. She has been a Principal Investigator for more than 30 national, provincial, and private grants and contracts. She is the Core Scientist of Yunnan Key Laboratory of Software Engineering and Yunnan Software

Engineering Academic Team, Kunming, China. Her research interests include blockchain, knowledge graph (KG), natural language processing (NLP), and recommendation system.



Xue Li received the B.E. degree in software engineering from Xi'an University of Technology, Xi'an, China, in 2020. She is currently working toward the M.E. degree with the School of Software, Yunnan University, Kunming, China.

Her current research interests include deep learning, computer vision, and WITMED.



Jishu Wang (Member, IEEE) received the B.E. degree in computer science and technology from Kunming University, Kunming, China, in 2019, and the M.E. degree in software engineering from Yunnan University, Kunming, China, in 2022, where he is currently working toward the Ph.D. degree with the School of Information Science and Engineering.

His current research interests include blockchain and intelligent transportation systems.



Rui Zhu (Member, IEEE) received the Ph.D. degree in software engineering from Yunnan University, Kunming, China, in 2016.

He is currently the Head and an Associate Professor of artificial intelligence with the School of Software, Yunnan University. His current research interests include intelligent transportation, blockchain technology, and deep learning.



Chen Gao (Graduate Student Member, IEEE) received the M.E. degree in software engineering from Yunnan University, Kunming, China, where he is currently working toward the Ph.D. degree with the School of Information Science and Engineering.

His research interests mainly include natural language processing and knowledge graph, especially information extraction. He has published three papers in the field of information extraction.



Qiuying Ma received the bachelor's degree in information security from Beijing University of Posts and Telecommunications, Beijing, China. She is currently working toward the master's degree with the School of Software, Yunnan University, Kunming, China.

Her main research interests are the construction of knowledge graph and entity extraction in the field of cyberspace security.