Information Processing and Management 60 (2023) 103412

Contents lists available at ScienceDirect

Information Processing and Management

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

Phishing scams detection via temporal graph attention network in Ethereum

Lei Wang [*], Ming Xu, Hao Cheng

College of Economics and Management, and Data Science and Systems Science (DTripleS) Lab, Nanjing Forestry University, Longpan Road 159, Nanjing 210037, China
Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies, Shenzhen 518055, China

ARTICLE INFO

Keywords:
Ethereum
Graph representation learning
Multilayer perceptron
Phishing scams detection
Temporal graph attention network
Temporal transaction information

ABSTRACT

Recently, phishing scams have become one of the most serious types of crime involved in Ethereum, the second-largest blockchain-based cryptocurrency platform. The existing phishing scams detection techniques for Ethereum mostly use traditional machine learning or network representation learning to mine the key information from the transaction network and identify phishing addresses. However, these methods typically crop the temporal transaction graph into snapshot sequences or construct temporal random wanderings to model the dynamic evolution of the topology of the transaction graph. In this paper, we propose PDTGA, a method that applies graph representation learning based on temporal graphs attention to improve the effectiveness of phishing scams detection in Ethereum. Specifically, we learn the functional representation of time directly and model the time signal through the interactions between the time encoding function and node features, edge features, and the topology of the graph. We collected a real-world Ethereum phishing scam dataset, containing over 250,000 transaction records between more than 100,000 account addresses, and divided them into three datasets of different sizes. Through data analysis, we first summarized the periodic pattern of Ethereum phishing scam activities. Then we constructed 14 kinds of account node features and 3 kinds of transaction edge features. Experimental evaluations based on the above three datasets demonstrate that PDTGA with 94.78% AUC score and 88.76% recall score outperforms the state-of-the-art methods.

1. Introduction

As the next generation of cryptocurrency and decentralized finance (DiFi) application platform, Ethereum is a major innovation and development of blockchain technique (Wang, Liu et al., 2021). It supports the creation of smart contracts to distribute applications and has the potential to become a virtual machine for the world of decentralized finance (Huang, Kong, Zhou, Zheng, & Guo, 2021; Wang, Cheng, Zheng, Yang and Zhu, 2021). Ether (ETH), the currency that underpins Ethereum, is now the second most valuable cryptocurrency by market capitalization, worth more than $ 220 billion. While the value is elevated, phishing scams in Ethereum are also on the rise (Chen, Pendleton, Njilla and Xu, 2020). Research organizations found more than 5000 phishing accounts in Ethereum, some of which have been active for the past two to three years. The stolen assets amount to upwards of tens of millions of dollars (Leng, Zhou, Zhao, Huang, & Bian, 2022). To protect the users and guarantee that the DiFi platform operates healthily, it is critical to detect phishing scams in the early stage for Ethereum transactions (Zheng, Xie, Dai, Chen, & Wang, 2018).

[*] Corresponding author at: College of Economics and Management, and Data Science and Systems Science (DTripleS) Lab, Nanjing Forestry University, Longpan Road 159, Nanjing 210037, China.
E-mail addresses: leiwangchn@163.com (L. Wang), xuxiaoming@njfu.edu.cn (M. Xu), 15150622539@163.com (H. Cheng).

https://doi.org/10.1016/j.ipm.2023.103412
Received 1 October 2022; Received in revised form 27 March 2023; Accepted 20 May 2023
Available online 31 May 2023

Unlike traditional phishing scams, which build fake platforms (websites or software) to collect sensitive information from victims or receive money transfers, phishing scams in Ethereum are phishers who use highly remunerative propaganda to induce remittances. Specifically, in the absence of fake platforms, attackers intentionally leak their private account keys through forums, chat rooms, etc. The common feature of phishing accounts is that there are ERC20 tokens that seem to be quite valuable, but the ETH that can be used to pay transaction fees is zero. The attacker wants to attract other users to transfer ETH to the phishing account as a transfer ERC20 Token fee. Attackers use scripts to monitor phishing accounts in real time, and once someone transfers ETH into it, the ETH will be transferred out immediately. For example, the account address 0xa8015df 1f65e1f53d491dc1ed3501 3031ad25034 is a phishing account. The attacker spent 30 ETH to buy 75,000 ERC20 tokens called ICX. The value of these 75,000 ICX tokens reached $320,000 at the peak, attracting countless victims who transferred ETH to the phishing account. Therefore, traditional phishing detection methods (Li, Yang, Chen, Yuan, & Liu, 2019) (mining fake platforms Williams & Polage, 2019 such as CSS styles Haruta, Asahina, & Sasase, 2018, website URLs Sahingoz, Buber, Demir, & Diri, 2019, etc.) are no longer suitable for Ethereum phishing scams.

A lot of research works have been proposed for Ethereum phishing scams detection. The earliest methods adopt traditional machine learning methods (Abdelhamid, Ayesh, & Thabtah, 2014). This kind of methods only consider simple statistical features such as the features in account or transaction attributes and in/out degrees to measure the similarity between accounts, while ignoring the topology of the ETH's transaction network, which may have a huge impact on the performance of phishing scams detection. Later, network embedding techniques became popular, such as DeepWalk, Node2Vec, etc. (Lin, Wu, Yuan, & Zheng, 2020). Some recent methods, such as graph convolutional networks (GCN), use some depth graph models to analyze the relationship of accounts (Wu et al., 2020). Although these methods perform passably on their own datasets, there are still several challenges for Ethereum phishing scams detection.

(1) Temporal dynamics of the Ethereum transaction graph need to be modeled as a function of continuous time. Relevant existing works (Li et al., 2022; Xie et al., 2021; Yu, Chen, Xu, & Xu, 2022) crop temporal graphs into snapshot sequences or construct temporal random wandering. Since time is a continuous variable, in order to model the dynamic and evolutionary properties of the Ethereum transaction graph, node embeddings are not only projections of topology structures and node features, but also functions of continuous time. Therefore, temporal representation learning should not only take place in the vector space but also in a certain function space.

(2) Temporal patterns exhibit in the dynamic evolution of the topology of the transaction graph, dynamic edge features and node features should be analyzed. Transactions that occurred long ago may have less impact on the current topology and therefore less impact on the node embedding. In contrast, more recent transactions have more influence on the current topology as well as the node embedding. In this regard, we use a self-attentive mechanism in this paper with temporal encoding to solve this problem.

(3) The interactions between temporal features should be well learned. Feature interactions are very effective in learning the regularities hidden behind the data. Some of the existing methods learn different types of features through different modules in the feature extraction process and finally stitching. The interactions between the dynamic evolution of the topology of the transaction graph, dynamic edge features and node features need to be well modeled.

The self-attention mechanism can help the model assign different weights to each part of the input, extract more critical information, and make the model perform more accurate judgment without bringing more calculation and storage overheads. Graph neural networks (e.g., GCN Welling & Kipf, 2016, GAT Velickovic et al., 2017) can efficiently aggregate node features and topological features, and may also further extend to incorporate edge features. Therefore, a graph neural network model with a temporal attention mechanism that can effectively aggregate dynamic node features and edge features makes sense that effectively detecting the phishing scams in Ethereum.

To deal with the above challenges, we propose PDTGA, a temporal graph attention network (TGAT)-based (Xu, Ruan, Korpeoglu, Kumar, & Achan, 2020) phishing scams detection approach for Ethereum. The core of the method is the combination of the self-attentive mechanism and the time encoding function. We model the temporal signal by the interactions between the temporal encoding function and the node features, edge features, and the topology of the graph, rather than simply concatenating different types of features after learning them through different modules. This makes our model more consistent with the explicit modeling of the temporality of nodes and edges as they are added, removed, or changed with continuous time, as required by the dynamic and evolutionary properties of the Ethereum transaction graph and learning the interactions between temporal features. We first count the information of each account and each transaction, and perform feature engineering on the obtained information to obtain node features and edge features as the input of the embedding layer. Then, PDTGA performs the node embedding process. The time-aware node embeddings are extracted through TGAT as the input for the node classification task. Finally, PDTGA utilizes multilayer perceptron (MLP) to detect phishing accounts. The main contributions of this work are summarized as follows.

- We propose an Ethereum phishing scams detection method, namely PDTGA, which can effectively aggregate dynamic structural features, edge features and node features to improve the detection performance of phishing scams in Ethernet transaction networks.
- We apply the self-attentive mechanism and the time encoding function to automatically learn feature interactions. It not only aggregates the temporal-topological neighborhood features but also learns the time-feature interactions. This trims the feature learning of the detection model more interpretable.
- Through data analysis methods including activity aggregation and transaction amount aggregation, we find that the weakness of Ethereum phishing scams is the periodicity of activities. Based on these, we summarize the activity characteristics of different stages in the life cycle of Ethereum phishing account, and we successfully construct temporal node features and edge features through feature engineering.

- We conduct extensive experiments on three Ethereum phishing scam datasets that we collected, and the results demonstrate that PDTGA has advantages over state-of-the-art approaches on multiple metrics.

The remainder of this paper is organized as follows. Section 2 summarizes the related works that significantly inspire our work. Section 3 introduces the preliminary knowledge. Section 4 presents PDTGA in detail, and Section 5 illustrates the experiments. Section 6 concludes the paper and points out some future research directions.

2. Related work

In this section, we first review the prior work on phishing scams detection in Ethereum transaction network, and then introduce the current mainstream methods of node classification, the core task of phishing account detection.

2.1. Ethereum phishing scams detection

Abnormal accounts detection is an important field of blockchain financial risk control research. Most studies in this field first perform feature extraction on accounts, and then use traditional machine learning methods for classification or clustering tasks. Kumar, Singh, Handa, and Shukla (2020) collect data from abnormal accounts and normal accounts in Ethereum and perform feature extraction, and classify them by combining traditional machine learning decision trees and other algorithms to identify abnormal accounts. Yin and Vatrapu (2017) use bagging and gradient boosting methods to evaluate illicit activity in Bitcoin in a supervised learning manner. Bian, Zhang, Zhao, and Shi (2020) extract manual features based on transaction history rules, and fuse them with statistical features to train the LightGBM model to detect malicious accounts in Ethereum. In order to improve the accuracy of anomaly detection, Jourdan, Blandin, Wynter, and Deshpande (2018) extract new features for constructing account profiles from the perspectives of account address, entity correlation, time, centrality, etc., and then use a decision tree model for classification.

In addition, deep learning methods such as graph representation learning can also be applied to the blockchain transaction network fraud detection by obtaining the low-dimensional space embedding vector of the account, that is, the deep account feature. For example, Wen, Xiao, Wang, and Wang (2023) use BP neural networks to extract the implied relationships between features and LSTM to capture the temporal features of transaction records. Sun, Ruan, and Liu (2019) use the Metapath2vec method to obtain the account depth features and then use the clustering method to detect abnormal accounts. Chen, Guo, Chen, Zheng and Lu (2020) and Chen et al. (2021) use graph neural networks, account cascade features and other methods to generate account features, and further utilize machine learning classification models to identify abnormal accounts. Li et al. (2022) use the LSTM model to obtain a temporal edge representation of the Ethereum transaction graph. The attention mechanism is used to aggregate edge representation around nodes to enrich node representation. The results are input into the GCN module to obtain structural features, and finally concatenate temporal edge features, structural features, and statistical features to identify phishing addresses. Xie et al. (2021) modeled Ethereum transaction records and linked consecutive snapshots. Random wandering learning node embeddings using a search strategy that relies on a number of transactions, structural transition probability, and temporal transition probability. Yu et al. (2022) consider node characteristics and structural characteristics, and uses timestamps as the weight of edges, arguing that the larger the timestamp (i.e., the more recent) the transaction occurred, the more critical it is.

Unlike the existing detection methods that model the dynamic evolution of the topology of the transaction graph by cropping the temporal graph into snapshot sequences or constructing temporal random wandering, we learn the functional representation of time directly in this paper. The temporal signal is modeled through the interactions between temporal encoding functions and node features, edge features, and the topology of the graph, rather than simply splicing after learning different types of features through different modules.

2.2. Node classification based on graph neural network

Node classification based on graph neural network is the core task of our method for detecting phishing accounts in Ethereum.

As for the graph neural network methods, Welling and Kipf (2016) propose a graph convolution network (GCN), which learns the node representation by encoding the first-order neighborhood structure, and comprehensively utilizes the network structure and node feature information to improve the node representation.

Since GCN cannot adapt to large-scale dynamic networks, Hamilton, Ying, and Leskovec (2017) propose a graph neural network framework, namely GraphSAGE (graph sample and aggregation), which samples the high-order neighborhood of the central node and aggregates neighborhood features by combining different pooling methods. Velickovic et al. (2017) believe that different neighbor nodes have different importance, and propose a graph attention network (GAT) to learn the weight of neighbor nodes, and weighted aggregation of neighbor features through multi-head attention to generate a more reasonable node representation. GAT obtains better classification results than GraphSAGE. Gilmer, Schoenholz, Riley, Vinyals, and Dahl (2017) propose a general message passing framework (message passing neural network, or MPNN), which generalizes the existing graph convolutional network, learns node representations through message passing rules and aggregation functions, and effectively models nodes' network information, such as relational features, and neighborhood structures.

A key reason for the success of many node classification methods is to explicitly exploit the connections between nodes. A particularly popular idea is to exploit its homogeneity. Based on the idea that nodes in a graph have similar properties to their neighbors, nodes have a tendency to share properties with their neighbors. For example, people prefer to be friends with people with the same interests or similar demographics. From the concept of homogeneity, machine learning models can be built that attempt to assign similar labels to adjacent nodes in the graph. In addition to homogeneity, there is the concept of structural equivalence, the idea that nodes with similar local structures will have similar labels. Furthermore, the notion of heterogeneity assumes that nodes will preferentially connect to nodes with different labels (e.g., gender attributes exhibit heterogeneity in many social networks). The relationship between phishing accounts and normal accounts is similar to this. Hence, this paper investigates phishing scams detection in Ethereum based on TGAT.

3. Preliminary knowledge

This section introduces a modification of the multi-head self-attention mechanism to adapt it for inductive representation learning on temporal graphs to identify and capture relevant segments in temporal neighborhood information.

3.1. Attention mechanism

The principle of attention mechanism is similar to the way human vision works. Since the ability to process information at the same time is always limited, people tend to selectively focus on the most important part of the information and ignore some unimportant information. Recently, attention mechanism has made remarkable achievements in modeling complex relationships. For example, it shows the advantages of modeling arbitrary word dependencies in machine translation and sentence embedding, and has been successfully applied to capture node similarity in graph embedding. We already know that we can classify nodes by their features like in-degree, out-degree, degree, and number of neighbors. The key problem is to determine which features should be combined to form meaningful higher-order features. According to the importance of the features, different weights are assigned to the features in the feature combination. Traditionally, this has been done by domain experts who create meaningful combinations based on their prior knowledge. In this paper, we use a novel approach to solve this problem, namely the multi-headed self-attention mechanism (Vaswani et al., 2017). Here, we extend this latest technique to model correlations between different characteristic fields. Specifically, we use the key–value attention mechanism to determine which combination of features is meaningful.

3.1.1. Self-attention mechanism

The attention mechanism introduced in this section refers to the scaled dot-product attention mechanism, which is defined as follows:

$$\mathrm{Attention}(Q,K,V)=softmax(\frac{QK^{\top}}{\sqrt{d_k}})V, \tag{1}$$

where Q and K are the input data with feature dimension d_k, and V is the input data with feature dimension d_v. According to the input data Q and K, the dot product can be obtained; then the weight corresponding to each element in the input data V can be obtained according to the softmax function. d_k is to scale the dot product to prevent the obtained dot product from being too large, which is conducive to the rapid progress of learning.

3.1.2. Multi-head self-attention mechanism

According to the scaled dot-product attention mechanism, a multi-head self-attention mechanism can be obtained. The essence of the multi-head self-attention mechanism is the linear transformation after the concatenation of multiple attention calculation results. This mechanism can make the model use different features obtained at different positions information, thereby increasing the diversity of features. Multi-head self-attention is calculated as

$$MultiHead(Q,K,V)=Concat(head_1,\dots,head_h)W^0, \tag{2}$$

where h represents the total number of heads, W^0 represents the weight matrix, $Concat$ represents the concatenation operation of the vector, and $head_i$ represents the feature of the ith head. $head_i$, which can be obtained according to the scaled dot product attention mentioned above, and the calculation method of head is

$$head_i=\mathrm{Attention}(QW_i^Q,KW_i^K,VW_i^V), \tag{3}$$

where W_i^Q represents the weight matrix when the ith head Q is used as input, W_i^K represents the weight matrix when the ith head K is used as input, and W_i^V represents the weight matrix when the ith head V is used as input.

3.1.3. Feed-forward network

The role of the feed-forward network (FFN) is to prevent the degradation of the model output (Dong, Cordonnier, & Loukas, 2021), which is mainly composed of two linear transformations using a $ReLU$ as the activation function. The feed-forward network is calculated as follows:

$$FFN(x)=max(0,\boldsymbol{x}W_1+b_1)W_2+b_2, \tag{4}$$

where W_1 and W_2 represent the weight matrix, b_1 and b_2 represent the bias terms, and max represents the operation of taking the maximum value.

3.2. Time encoding

The self-attention mechanism uses positional encoding, that is, for the input entity embedding (or feature) ($Z \in \mathbb{R}^d$), a corresponding position vector p_k is concatenated to learn the features containing positional relationship as

$$Z = [z_1 \| p_1, \dots, z_i \| p_l]^\top \in \mathbb{R}^{l\times(d+d_{pos})}. \tag{5}$$

Since the self-attention mechanism only captures sequence information through positional encoding, it cannot handle temporal features. And since time is a continuous variable, the mapping from time domain to vector space must be functional. Therefore, it is necessary to replace the position encoding in Eq. (5) with a continuous function mapping $\boldsymbol{\phi} : T \to \mathbb{R}^{d_T}$ from the time domain to the d_T dimensional vector space. The temporal signal is subsequently modeled by the interactions between functional temporal encoding and node features and graph topology.

The time function (Xu, Ruan, Korpeoglu, Kumar, & Achan, 2019) is used to encode the relative time difference between source (SRC) node and destination (DST) node as

$$\boldsymbol{\phi}(t, t_i) = (cos(\omega_1(t - t_i) + b_1), cos(\omega_2(t - t_i) + b_2), ..., cos(\omega_d(t - t_i) + b_d)) \in \mathbb{R}^d. \tag{6}$$

So Z can be transformed into:

$$Z = [z_1 \| \boldsymbol{\phi}(t, t_1), \dots, z_i \| \boldsymbol{\phi}(t, t_i)]^\top \in \mathbb{R}^{l\times(d+d_T)}. \tag{7}$$

The reason for encoding with relative time difference $|t - t_i|$ is due to the following considerations:
(1) Relative time is intuitively more important than absolute time;
(2) Time difference has translational invariance, which is convenient for discovering periodic laws;
(3) In essence, it is necessary to learn the kernel function:

$$\mathrm{K}(t_1, t_2) = \langle \boldsymbol{\phi}(t_1), \boldsymbol{\phi}(t_1) \rangle = \boldsymbol{\phi}(t_1, t_2). \tag{8}$$

Eq. (8) uses the cosine function for coding, because the cosine function is periodic and can reflect the periodic change of the time difference. Different ω represent different frequencies, and d-dimensional features can extract different frequencies.

4. PDTGA approach

4.1. Problem statement

In this paper, the Ethereum phishing scams detection task is formulated as a graph node classification problem. Let an Ethereum transaction network be $G = (V, E)$, where V represents the node set, that is, the Ethereum address set, and E represents the edge set in the transaction network, that is, the Ethereum transaction record. $G_L = (V, E, X, Y)$ is a transaction network with labels, each node has a feature vector, and the node feature matrix is $X \in \mathbb{R}^{|V|\times d}$, where d is the dimension of each node feature vector; the label matrix $Y \in \mathbb{R}^{|V|\times Y}$, where Y is the set of labels. The classifier $f : V \mapsto Y$ is the resulting model trained on dataset $\mathbb{D}$ and used to predict the behavior type of Ethereum addresses. In the scenario of phishing fraud transaction recognition, Y contains two kinds of labels, namely $y \in Y = \{1, 0\}$. In this paper, 1 is used to represent the sample label of suspected phishing fraud activities, and 0 is used to represent the label of normal and legitimate transaction samples. Therefore, when $f(x) = 1$, sample x is classified as an abnormal transaction. Otherwise, sample x is classified as a normal transaction. As shown in Fig. 1 of the transaction graph example, a_0, a_1, a_2, a_3, a_4, represent different Ethereum accounts, and the directed edge between each node indicates that there is a transaction record between the two Ethereum accounts. The transaction record is that the source node transfers a certain amount of ETH to the destination node. For example, Ethereum account a_1 transfers 2 ETH to Ethereum account a_0 at time point $t = 1527681178$. Depending on the above method, this paper builds a directed transaction network graph based on the Ethereum transaction record, and uses the adjacency matrix to represent and store it.

4.2. Overview of PDTGA

PDTGA method is a semi-supervised classification method, and the node embedding vector obtained through graph representation learning can be directly used as the input of the downstream node classification model to train the classification model, so as to classify the nodes of unknown classes. Since most of the time information is reflected through the timely interactions between nodes, it is necessary to effectively aggregate temporal node features and temporal edge features, and assign different weights to each feature according to its contribution to phishing account detection. We do this with a temporal graph attention layer in this paper. Node classification is then treated as the downstream task using the obtained time-aware node embeddings as input.

The flow of the entire architecture is illustrated in Fig. 2. It involves the following five steps.
(1) Construct temporal transaction graph and sample subgraph;
(2) Data analysis and feature engineering;
(3) Embedding layer;
(4) Downstream node classification;
(5) TGAT Model Training.

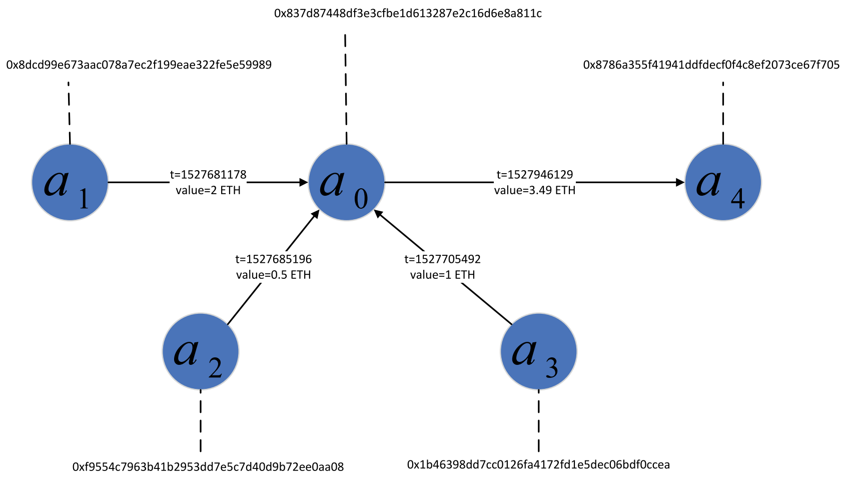

0x837d87448df3e3cfbe1d613287e2c16d6e8a811c
0x8dcd99e673aac078a7ec2f199eae322fe5e59989
0x8786a355f41941ddfdecf0f4c8ef2073ce67f705
a_1
a_0
a_4
a_2
a_3
t=1527681178 value=2 ETH
t=1527946129 value=3.49 ETH
t=1527685196 value=0.5 ETH
t=1527705492 value=1 ETH
0xf9554c7963b41b2953dd7e5c7d40d9b72ee0aa08
0x1b46398dd7cc0126fa4172fd1e5dec06bdf0ccea

Fig. 1. The Ethereum transaction graph example.

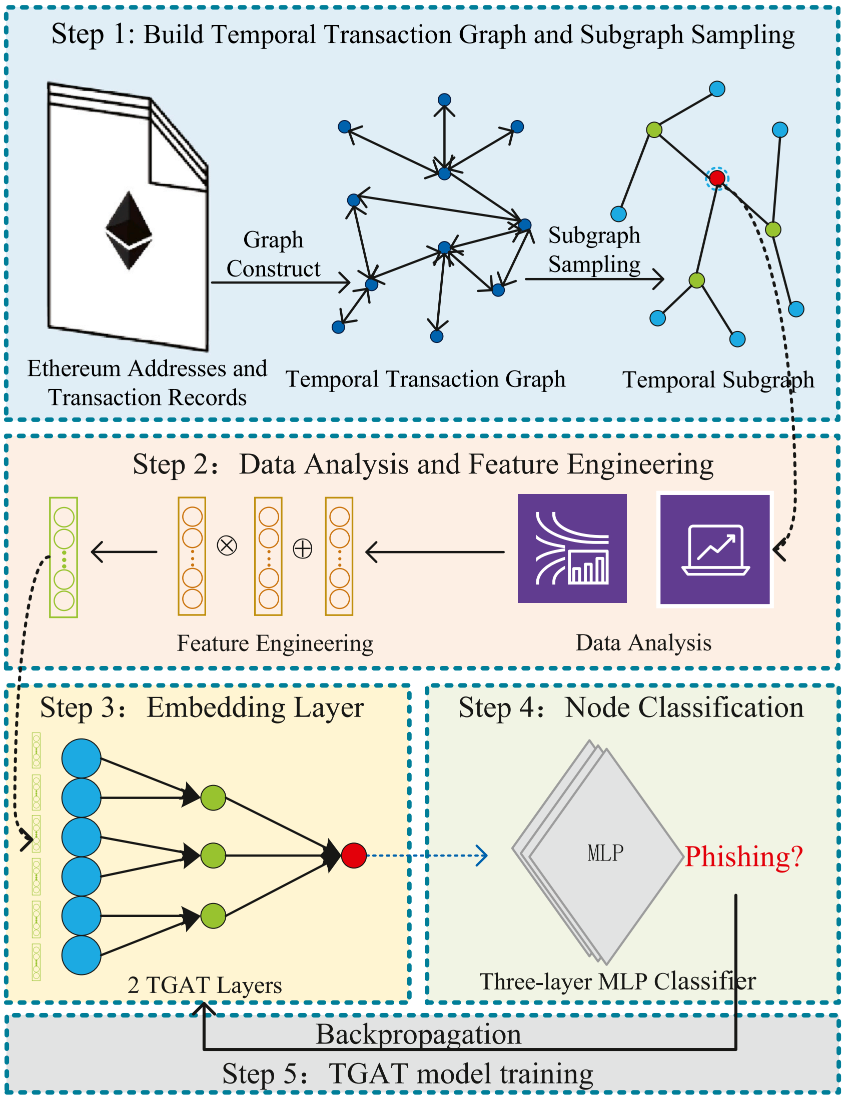

Step 1: Build Temporal Transaction Graph and Subgraph Sampling
Graph Construct
Subgraph Sampling
Ethereum Addresses and Transaction Records
Temporal Transaction Graph
Temporal Subgraph
Step 2: Data Analysis and Feature Engineering
Feature Engineering
Data Analysis
Step 3: Embedding Layer
2 TGAT Layers
Step 4: Node Classification
MLP
Phishing?
Three-layer MLP Classifier
Backpropagation
Step 5: TGAT model training

Fig. 2. The overall architecture of PDTGA.

4.3. Construct temporal transaction graph and sample subgraph

Based on massive transaction data crawled from Ethereum, we first construct a large-scale Ethereum transaction graph. In the transaction graph, we use nodes to represent accounts and edges to represent transactions between accounts. Note that the Ethereum transaction graph allows any pair of nodes to have multiple directed edges, and each transaction edge carries three types of information: transaction amount, transaction timestamp and transaction direction. The algorithm for constructing temporal transaction graphs is presented in Algorithm 1.

```
Input: Ethereum transaction set $E_t$
Output: Ethereum transaction graph $G$
1 Function Constructing Temporal Transaction Graph($E_t$):
2     $G \leftarrow$ empty graph; $V \leftarrow$ empty set of nodes;
3     for each transaction e in $E_t$ do
4         add a directed edge $e$ ($v.src \rightarrow v.dst$) with transaction amount and timestamp to edge set $E$ in $G$;
5         add unique $v.src$ and $v.dst$ to node set $V$;
6     end
7     return $G$;
```

Algorithm 1: Construct Temporal Transaction Graph.

The original transaction graph is large, so we take a random walk method to sample. We start walking by randomly selecting a node from the graph, then randomly selecting its neighbors as the next node, and repeating this process until the number of neighbor nodes satisfies our setting. Then we get a transaction subgraph of the scale we want. The algorithm for sampling temporal subgraph is presented in Algorithm 2.

```
Input: Ethereum transaction graph $G$, starting node $s$, number of neighbors $k$
Output: Ethereum transaction subgraph $G'$
1 Function Sample Transaction Subgraph($G, s, k$):
2     $G' \leftarrow$ empty subgraph; $V' \leftarrow s$;
3     while $|V'| \leq k+1$ do
4         $n \leftarrow$ a random node in the neighbors of $s$;
5         add node $n$ and directed edge between $s$ and $n$ to node set $V'$ and edge set $E'$ in $G'$;
6     end
7     return $G'$;
```

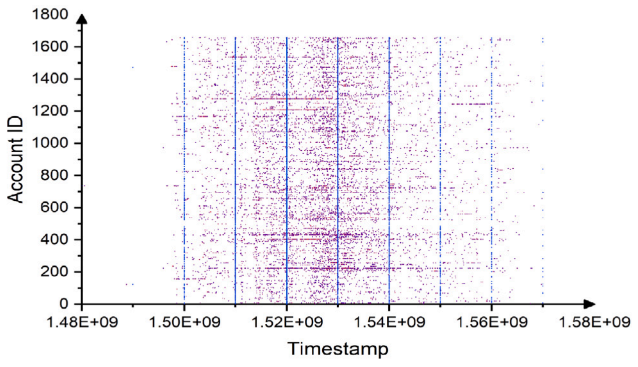
Algorithm 2: Sample Temporal Subgraph.

4.4. Data analysis and feature engineering

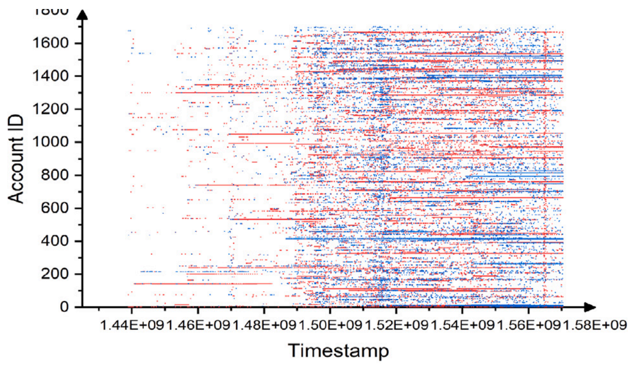
Fig. 3 illustrates the behavior pattern of each account over time, where each dot represents a transaction. In particular, each blue dot indicates that the account i has an outgoing transaction at time t, and each red dot represents account i has an incoming transaction at time t. The behavior of the normal accounts in Fig. 3(b) demonstrates that each newly registered normal account behaves evenly over the following time period, while the phishing accounts in Fig. 3(a) show that they tend to only explode for a short period of time and then go quiet. After statistics, the average active life cycle of these phishing accounts is 67 days and 8 h.

As can be seen from Fig. 3(b), the time distribution of external transfers and internal transfers of normal accounts is uniform, and there is no significant difference. Part of external transfers of phishing accounts is often concentrated near certain time nodes. We believe that this is because phishing attackers simultaneously hold a large number of accounts used for phishing scams, transferring the fraudulently obtained ETH to secure accounts or laundering money at certain times. So if there is a batch of accounts that always transfers money around the same time node, we consider this batch of accounts suspicious. The inner product of the activities of two accounts sharing the same activity time series can be simply defined as a measure of affinity, namely $S_{i,i'} = \langle x_i, x_{i'} \rangle$. Clearly, consistent behavior over time between accounts i and i' implies a high degree of affinity. This affinity measure between two accounts can be further used to segment a huge connected subgraph to increase the false positive rate (Zhao et al., 2009).

Fig. 4 illustrates the distribution of the amount of transactions that each account participated in. In particular, each dot represents a transaction. Each blue dot indicates that the account i participated in a transaction with an amount s. Comparing the distribution of transaction amounts between normal accounts and phishing accounts in Fig. 4(a) and (b), it can be clearly seen that phishing accounts prefer small-value transactions compared to normal accounts. Statistically, the median transaction amount for phishing accounts is 0.017621 and the median transaction amount for normal accounts is 3.46622. This should be related to the pattern of phishing scams, where phishers use high-reward propaganda to induce remittances, defrauding the transfer of transaction fees, which are often modest.



(a) Phishing account



(b) Normal account

Fig. 3. Activity aggregation mode: phishing accounts (a) v.s. normal accounts (b). Blue dots represent external transfers, and the red dots represent internal transfers. Newly registered phishing accounts are often only active in a short period of time. Part of external transfers from phishing accounts is often concentrated around a few fixed time nodes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Based on the above comparative analysis of the behavior patterns of phishing accounts and normal accounts, we can roughly divide the Ethereum phishing scam activities into the following three periods.

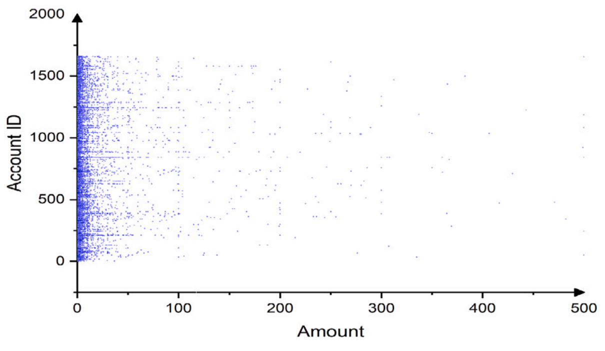
(1) Dissemination period: During this period, phishers spread phishing scam information as much as possible. Phishing scams can be spread quickly over the Internet. Although the process of spreading phishing scams is not directly reflected in transaction records making it difficult to measure the time spent, and the spread period is usually very short.

(2) Spring period: Phishing scam information in this period has been fully spread. A large number of victims are fooled into transferring money to phishing accounts. Subsequently, the frequency of transactions drops dramatically. Suppose that the time it takes for an account to complete half of the total number of transactions is considered the spring period. We randomly sampled the transaction records of 1500 labeled accounts on the Etherscan website for statistical analysis. The results show a mathematical expectation of 13.64% for the spring period as a percentage of the whole account active period, with a standard deviation of 0.1643. Specifically, we consider the time between the first and the last transactions of the account as the account active period.

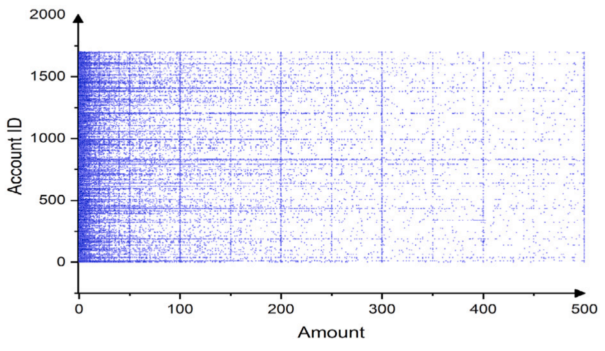
(3) Quiescent period: During this period, on the one hand, the phishing scam lures posted earlier have sunk on the Internet over time, and the number of swindled victims has dwindled. On the other hand, as more and more victims report, which triggers warning messages from the platform, fewer and fewer victims become victims. As a result, the number of transactions transferred to phishing site accounts has dropped significantly and the frequency has stabilized. At the same time, phishers transfer balances out of their accounts for money laundering. We consider the account active period other than the spring period as the quiescent period, and the mathematical expectation of its share is 86.36% with a standard deviation of 0.1643.

The periodic behavior rule of phishing account trading above is an essential basis for detecting phishing accounts. Phishing scams by a phishing account will further affect its neighbor nodes, making them behave differently from other normal accounts' transaction behavior patterns. This prompted us to consider modeling phishing accounts from a graph perspective.

Next, we perform feature engineering on the collected subgraph information in preparation for the subsequent learning of node embeddings. Due to the anonymity of the blockchain platforms, the nodes themselves do not carry any portrait information. Feature engineering can only be achieved by extracting and combining basic information in the relationship between nodes. Therefore, based on the above analysis of the differences between phishing accounts and normal account transaction behavior patterns, we extract the following 14 dimensional features as attribute features of nodes. They are: (1) the total degree of the node; (2) the out-degree; (3) the



(a) Phishing account



(b) Normal account

Fig. 4. Transaction amount aggregation mode: phishing account (a) v.s. normal account (b), phishing accounts are more inclined to small-amount transactions than normal accounts.

in-degree; (4) the ratio of in-degree; (5) the ratio of out-degree; (6) the sum of transaction amounts; (7) the transfer-out transaction amount; (8) the transfer-in transaction amount; (9) the difference between transfer-in and transfer-out transaction amount; (10) the transfer-in-transfer-out amount ratio; (11) the ratio of transfer in neighbors; (12) the ratio of transfer out neighbors; (13) the total number of neighbors; (14) the inverse of the transaction frequency (Hu et al., 2021). As for edge features, we select the transaction amount, transaction direction, and the timestamp of the transaction (Estevam, Palma, Silva, Martina, & Vigil, 2021).

4.5. Temporal graph attention layer

Our proposed PDTGA method mainly relies on the temporal graph attention layer (TGAT layer), which can naturally achieve the aggregation of node features, edge features, and temporal features. As illustrated in Fig. 5, the TGAT layer uses $\boldsymbol{a_i} \in \mathbb{R}^{d_0}$ to represent the raw node feature vector of node i. For the dynamic edge of the temporal graph, we use $\boldsymbol{e_{i,j}(t)}$ to represent the edge feature vector resulting from the interaction between nodes i and j at time t (Wang et al., 2019).

The TGAT layer is similar to the GAT model and can be used as a local aggregation operator, which takes the hidden representation or feature vector of the DST node and its temporal neighborhood, the feature vector of the edge between the DST node and the temporal neighborhood, and the timestamp as input. The corresponding output is the time-aware representation of the DST node at the time t.

We define the hidden representation of node i at time t in layer l as $\vec{h}_i^{(l)}(t)$. For DST node 0, its SRC node set is $\mathbb{N}(0;t) = \{1,\ldots,n\}$. The interactions between nodes 0 and $i \in N(0;t)$ occurs at time t_i. Then the input Z of the TGAT layer is the neighborhood information of the DST node $\{\vec{h}_1^{(l-1)}(t),\ldots,\vec{h}_n^{(l-1)}(t_n)\}$ and the interaction information between the SRC node and the DST node $\boldsymbol{e_{i,j}(t)}$, and the information of the DST node itself $(\vec{h}_0^{(l-1)}(t),t)$. We set the number of TGAT layers to 2, i.e. $l = 2$, and for the 2^{th} layer, the input is just raw node features, edge features and timestamps. The DST node information at these time points is expressed as $(\vec{h}_0^{(l-1)}(t),t)$. We define the time-aware representation of the DST node 0 generated by this layer at time t as $\vec{h}_0^{(l)}(t)$. The kernel function can be defined as $\boldsymbol{\phi}(t-t_i)$, because of the translation invariance assumption of the time kernel, and we only care about the time span, then $t-t_i$ can be used as interactive time. In this way, according to our modification of the self-attention mechanism in Section 3.2, we can obtain the feature matrix : $Z(t) = [\vec{h}_0(t)\|\boldsymbol{e_{0,0}}\|\boldsymbol{\phi}(0), \tilde{\mathrm{h}}_1(\mathrm{t})\|\boldsymbol{e_{0,1}}\|\boldsymbol{\phi}(\mathrm{t}-\mathrm{t}_1),\ldots,\tilde{\mathrm{h}}_\mathrm{n}(\mathrm{t})\|\boldsymbol{e_{0,n}}\|\boldsymbol{\phi}(\mathrm{t}-\mathrm{t}_\mathrm{n})]$. Next, we perform three different linear projections on Z to obtain 'query', 'key' and 'value':

$$Q(t) = [Z(t)]_0 = (\vec{h}_0(t)\|\boldsymbol{e_{0,0}}\|\boldsymbol{\phi}(0))W_q, \tag{9}$$

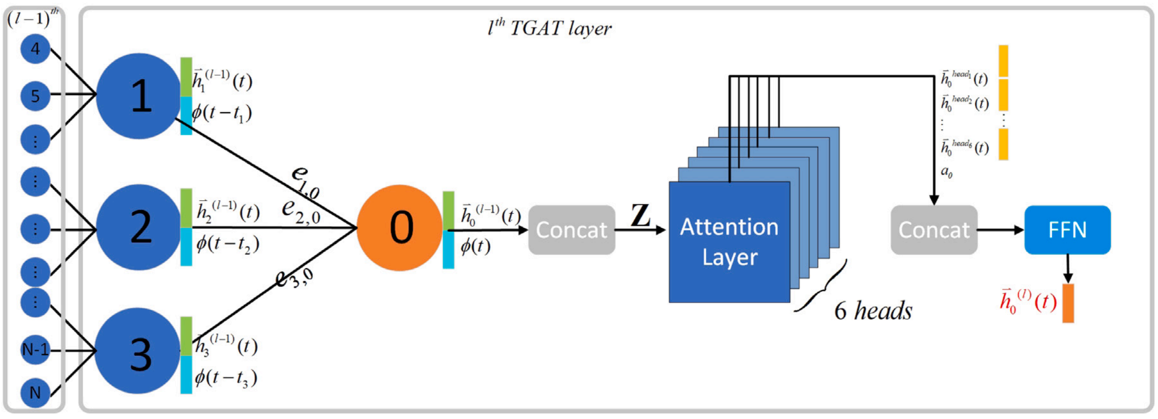


Fig. 5. The framework of the l^{th} TGAT layer with $h = 6$ attention heads for node 0 at time t.

$$K(t) = [Z(t)]_{1:n} = (\vec{h}_1(t_1)\|\boldsymbol{e_{0,1}}\|\boldsymbol{\phi}(t-t_1), \dots, \vec{h}_n(t_n)\|\boldsymbol{e_{0,n}}\|\boldsymbol{\phi}(t-t_n))W_k, \tag{10}$$

$$V(t) = [Z(t)]_{1:n} = (\vec{h}_1(t_1)\|\boldsymbol{e_{0,1}}\|\boldsymbol{\phi}(t-t_1), \dots, \vec{h}_n(t_n)\|\boldsymbol{e_{0,n}}\|\boldsymbol{\phi}(t-t_n))W_v. \tag{11}$$

where $W_Q, W_K, W_V \in \mathbb{R}^{(d+d_e+d_T)\times d_h}$ are the weight matrices that need to be learned to capture the interactions between time encoding, node features and edge features.

According to Eq. (1), we take the dot product self-attention output as a neighborhood hidden representation:

$$\vec{h}_0(t) = softmax(\frac{QK^{\top}}{\sqrt{d_k}})V. \tag{12}$$

The hidden neighborhood representation is subsequently concatenated with the feature vector $\boldsymbol{a_0}$ of the target node and passed to a feed-forward network (FFN).

$$\vec{h}_0^{(l)}(t) = FFN(\vec{h}_0^{head_1}(t)\|\vec{h}_0^{head_2}(t)\|\dots\|\vec{h}_0^{head_h}(t)\|\boldsymbol{a_0}). \tag{13}$$

4.6. Phishing accounts classify

The task of this module is to classify nodes to distinguish phishing nodes from normal nodes. After the above operations, we get the temporal node and edge features, and combine them together as a complete node representation. On the basis of obtaining the complete node representation, we need to learn the difference between phishing and normal node representation. So we need a classifier that can dig deep into the differences between fishing and non-fishing node representations to classify Ethereum phishing accounts. We employ a three-layer MLP as a classifier to classify phishing and non-phishing accounts with a time-aware representation as input. We use this classifier for all the comparison methods in Section 5 under all three datasets for the phishing addresses classification step.

4.7. TGAT model training

Sections 4.5 and 4.6 can be cast as a process of forward propagation. To guarantee the effectiveness of the model for phishing scams detection, we also present the process of backpropagation in this section. For the process of backpropagation, the BCELoss (binary cross entropy loss) function with weight parameters is used to address the class imbalance between phishing accounts and normal accounts (Fan, Fu, Xu, & Cheng, 2021; Zeng, Li, Xiao, Shen, & Zhong, 2022). In addition, we used the Adam optimizer to update the model parameters. Specifically, during the training process, the gradient of the loss function calculated by the backward function is passed to the Adam optimizer, which calculates the update amount of the model parameters and uses it to update them. As a result, the value of the model's loss function continuously decreases, making the model's predictions more accurate. The training process of TGAT model forward propagation and backpropagation is summarized in Algorithm 3.

5. Experiments

In this section, we perform experimental evaluations to demonstrate the effectiveness of PDTGA. Specifically, we aim to answer the following research questions.

```
Input: Neighborhood information $Z(t)$ of the target node
Output: The trained TGAT Model
for trained epoch ≤ the specified number do
    //Forward propagation
    //Calculate the representation vector of each node through the TGAT layer
    $\vec{h}_0^{(l)}(t)$ = FFN(MultiHeadAttention($Z(t)$) );

    // Output layer
    $output$ = MLP($\vec{h}_0^{(l)}(t)$);

    //Output layer calculation loss function
    $loss$ = BCELoss($output$, $y$), $y$ is the node label;

    //Backpropagation
    //Calculate gradients
    $gradients$ = backward($loss$);

    //Update model parameters AdamOptimizer.update($gradients$);
end
return The trained TGAT Model;
```

Algorithm 3: TGAT Model Training Algorithm.

- RQ1: How effective is the proposed PDTGA method for detecting phishing accounts on the Ethereum transaction network?
- RQ2: How does each kind of feature contribute to the final detection performance? And the contribution of attention aggregation?
- RQ3: By providing hyperparameters such as different attention sizes, number of embedding layers, etc., how much will the classification performance of PDTGA methods change?

5.1. Dataset

5.1.1. Dataset description

We conducted the experiments on a partial second-order transaction network dataset of Ethereum. We downloaded the original dataset from the well-known XBlock[1] website. This dataset takes the accounts labeled as phishing and some unlabeled accounts in Ethereum as the central node, and uses the API provided by Etherscan[2] to extract their first-order, second-order neighbors and transactions between them. The dataset is divided into two parts: (1) the first-order transaction network dataset, and (2) the second-order transaction network dataset. The first-order transaction network dataset contains 1660 phishing target nodes and 1700 non-phishing target nodes crawled from Etherscan. The transaction information between the target node and its first-order neighbor nodes is recorded in a csv file. The csv file for the second-order transaction network dataset also records the transaction information between each first-order neighbor node and its corresponding second-order neighbor node. Each line in the CSV file represents a transaction information that the target node participates in. Each piece of transaction information includes: (1) *TxHash*, i.e., transaction hash value; (2) *BlockHeight*, i.e., the block height of the exchange; (3) *TimeStamp*, i.e., transaction timestamp; (4) *From*, i.e., transaction originating node address; (5) *To*, i.e., transaction receiving node Address; (6) *Value*, i.e., transaction amount. Due to the large scale of the complete transaction graph, we sampled transaction graphs containing 100, 150, and 200 labeled nodes, and used the transaction records between nodes on the transaction graphs as the experimental datasets. After data pre-processing, we recorded them as D_1, D_2, D_3, respectively.

5.1.2. Data pre-processing

For labeled target nodes, we eliminate the nodes with more than 500 transactions or less than 5 transactions (Chen, Guo et al., 2020; Wu et al., 2020). For nodes with too many transactions, we think it could be a wallet or other non-phishing account. This represents only 0.5% of the total number of nodes, so we consider the nodes labeled as phishing to be reliable. If the number of labeled target node transactions is too small, we consider it meaningless in training. At the same time, we eliminated a very small number of transaction records with missing information. After data cleaning, the number of remaining nodes in each dataset is 29,539, 33,720, and 39,854, respectively. More detailed information of the datasets is shown in Table 1.

[1] http://xblock.pro/.
[2] https://etherscan.io.

Table 1
Statistics of the datasets.

Subgraph	Labeled nodes	Edges	Average degree
D_1	100	1,971,284	66.7350
D_2	150	2,047,276	60.7140
D_3	200	2,264,986	56.8321

Finally, the pre-processed data is fed into the TGAT layer to obtain node representations for downstream classification tasks. As shown in Fig. 4(a), the distribution of phishing labels appears inhomogeneous over time. In addition, the PDTGA model proposed in this paper relies on mining transaction features and structural features between nodes and their neighboring nodes, and the dynamic evolution features of these features over their life cycle. If the time span of the test set is made too small, just a very small segment of the life cycle, it is difficult to mine the features that distinguish phishing nodes from non-phishing nodes, leading to false alarms. Therefore, in the classification task, we split the data based on the temporal order of the observations, earlier 70% data is split as training set, and the remaining 30% data is test set.

5.2. Experimental settings

5.2.1. Comparison methods

We compare PDTGA with 3 classes of state-of-the-art Ethereum phishing scams detection methods, including: (1) Feature-based methods that only consider node attributes; (2) Random walk-based network embedding methods (i.e., DeepWalk, Node2Vec and LINE); (3) Graph Embedding based on graph neural network (i.e., GCN, GraphSage, GAT and TTAGN). The following eight representative approaches are compared in this paper.

- **Feature-only.** The 14-dimensional feature vectors of the node are fed directly into the classifier.
- **DeepWalk.** The core idea of DeepWalk (Welling & Kipf, 2016) applied to node classification is mainly divided into two parts, namely random walk and vector representation of generated nodes. In the random walk process, the algorithm first needs to extract the node sequence from the network graph, and uses the Word2vec to treat the node sequence as a sentence composed of a single word in the Word2vec algorithm. The representation is then generated as a vector with d dimensions.
- **Node2Vec.** Node2Vec (Grover & Leskovec, 2016) is a graph embedding method that comprehensively considers DFS neighborhoods and BFS neighborhoods. In simple terms, it can be regarded as an extension of Deepwalk, which can be regarded as a Deepwalk that combines DFS and BFS random walks.
- **LINE.** DeepWalk is a DFS-based algorithm, so it focuses on exploring second-order similarities between nodes, i.e., community discovery. The most intuitive first-order similarity modeling of nodes in the graph is missing. In response to this defect, the idea of LINE (Tang et al., 2015) is to combine first-order similarity and second-order similarity to find node representation.
- **GraphSAGE.** GraphSAGE (Hamilton et al., 2017) performs a linear transformation on the attribute features of the node itself and the sampled neighbor node features, and then combines the two. It performs a linear transformation to obtain the feature representation of the target node, and the obtained target node representation can be used for the downstream node classification task.
- **GAT.** GAT (Velickovic et al., 2017) aggregates neighbor nodes through attention mechanism to achieve adaptive allocation of weights to different neighbors. The weights of different neighbors in GCN are fixed and come from the normalized Laplacian matrix. Thus, GAT greatly improves the representation ability of the graph neural network model.
- **GCN.** GCN was first applied in Chen, Guo et al. (2020) and Chen et al. (2021) for phishing fraud detection in Ethereum. GCN considers the characteristics of adjacent nodes and updates the node representation vector of each node in the network graph, which improves the effectiveness of clustering.
- **TTAGN.** TTAGN is an Ethereum phishing detection method recently proposed in Li et al. (2022). It uses an LSTM model to obtain temporal edge representations. Attention mechanism is used to aggregate edge representations around nodes to enrich node representations. GCN module is used to obtain the structural features.

5.2.2. Implementation details

In this section, we present the experimental details of this paper.

For our proposed method, we search the number of TGAT layers in {1,2,3} and the number of attention heads in {1,2,3,4,5}. Although our method does not limit the neighborhood size during aggregation. To speed up training when using multi-hop aggregation, we use uniformly sampled neighborhood dropout (chosen p = {0.1, 0.3, 0.5}). During training, we tune the initial learning rate in {0.1,0.01,0.001} and use Glorot initialization and Adam SGD optimizer to optimize our model. Subsequent experiments will demonstrate that a two-layer TGAT layer with an attention size of 6, a learning rate of 0.01, and a dropout of 0.1 appears the best performance. To prevent overfitting, flood regularization is used on the loss function (ao Huang, Sang, Sun, & Lv, 2022). Considering the imbalance of positive and negative sample data, we set the weight parameter W of the BCELoss function to 30 in subsequent experiments (Li et al., 2021; Wu, Gu, & Gu, 2017). The embedding size for all experiments is set to 50, and the batch size is set to 3000, considering the training speed and computer configuration. We apply an early stopping strategy in all training processes. If the F1 score does not improve (tolerance is 0.001) after 3 epochs, we stop training. A random seed is set to

Table 2
Performance comparison results w.r.t. AUC, Recall, Precision and F1 score on three datasets. The highlighted results present the highest performance.

Method	D_1				D_2				D_3			
	AUC	Recall	Precision	F1 score	AUC	Recall	Precision	F1 score	AUC	Recall	Precision	F1 score
Features only	0.735	0.3977	0.7180	0.5119	0.7204	0.3252	0.6320	0.4294	0.7131	0.3809	0.6467	0.4794
DeepWalk	0.7334	0.5261	0.7142	0.6059	0.6921	0.5734	0.4487	0.5035	0.7377	0.3894	0.5904	0.4693
Node2Vec	0.6483	0.4155	0.6024	0.4918	0.6845	0.5506	0.4360	0.4866	0.8000	0.6614	0.4948	0.5661
LINE	0.7933	0.7154	0.6431	0.6773	0.7686	0.6356	0.6433	0.6394	0.8218	0.6106	0.6169	0.6137
GAT	0.7020	0.7340	0.8010	0.7660	0.7260	0.7750	**0.8225**	0.7980	0.7340	0.7630	0.8310	0.7955
GraphSAGE	0.6720	0.8505	0.8175	0.8337	0.6550	0.7735	0.8070	0.7899	0.6540	0.7945	0.8225	0.8083
GCN	0.7128	0.5695	0.6265	0.5966	0.7884	0.7009	0.7589	0.7288	0.7530	0.6518	0.6325	0.6420
TTAGN	0.8849	0.7666	0.8179	0.7914	0.8918	0.7509	0.8163	0.7822	0.9074	0.7415	**0.8318**	0.7841
PDTGA	**0.9251**	**0.8513**	**0.8218**	**0.8363**	**0.9377**	**0.8736**	0.8068	**0.8389**	**0.9478**	**0.8876**	0.8015	**0.8423**

ensure consistent experimental results. For DeepWalk and Node2Vec, walk length and window size are set to 20 and 4, and p and q for the latter are set to 0.25 and 0.4, respectively.

Our model relies on the Pytorch framework, the network embedding baseline relies on the OpenNE open source package, and the Graph Neural Network model is implemented through the PyG framework.

5.2.3. Evaluation metrics

We employ the following metrics to evaluate the effectiveness of different approaches.

- Precision. The precision rate means the percentage of real phishing nodes that are judged as phishing nodes.
- Recall. The recall rate refers to the percentage of phishing nodes that are judged to be true phishing nodes.
- F1 score. F1 score is the harmonic mean of the Precision and Recall score.
- Area Under Curve (AUC). The AUC metric is to calculate the area under the ROC curve formed by true positive rate (TPR) and false positive rate (FPR) with multiple thresholds, which is frequently used in binary classification tasks.

During the performance evaluation, we pay more attention to AUC score and Recall score. The AUC score is the basis for judging the overall validity of the model. The Recall score refers to the proportion of detected phishing nodes to real phishing nodes. In general, mislabeling a normal account may have a serious impact on the reputation of the platform itself, but if a fraudulent account is missed, the user can incur a significant capital loss. The focus of the model on the Precision score and Recall score may vary for different business scenarios. If the crypto project needs to expand its business, in order to reduce the false catch rate of good accounts and ensure that it can attract more users, the risk control department will raise the threshold value, which will increase the model's detection rate of Precision value and, at the same time, it will cause the detection rate of Recall to decrease. If the crypto project wants to tighten its business to catch as many bad accounts as possible, the risk control department will lower the threshold value, thus increasing the model's check-all rate of Recall, but this will cause some good accounts to be mis-caught, thus decreasing the model's check-all rate of Precision. Even if some normal accounts are labeled as high-risk accounts, they can do a second review in the subsequent manual review. However, it is very difficult to recover the loss of fraudulent accounts. In the modeling exercise, we inevitably have to make a trade-off between Precision score and Recall score. When the F1 Score is comparable, we prefer the model with a higher Recall score.

5.3. Effectiveness evaluation (RQ1)

We evaluate the performance of all the comparison methods mentioned above in the phishing scams detection task, and summarize the four evaluation metrics scores of each method as shown in Table 2.

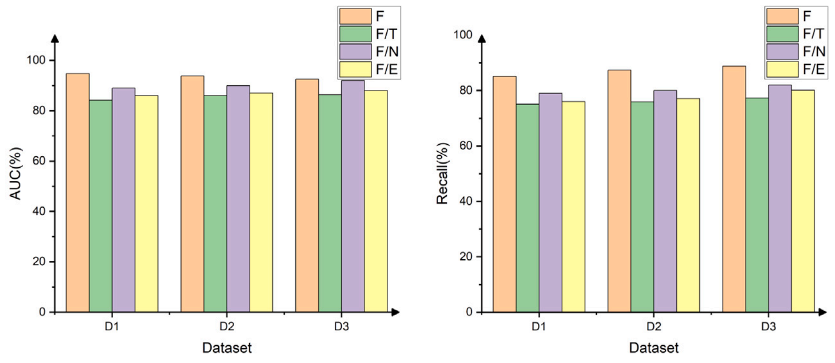
As can be seen from Table 2, we can draw the following conclusions:

(1) The combined performance of the PDTGA method was optimal. The metric of AUC gives a good overview of the performance of the imbalanced sample classifier. The dataset we face in this paper is extremely unbalanced, in the real Ethereum in which there are far fewer fishing nodes than non-fishing nodes. Our method achieves a significant advantage over other baselines in terms of AUC scores on datasets of different sizes. Even compared to the state-of-the-art TTAGN method, we still have an advantage of over 4 percentage points on each dataset. GraphSAGE is comparable to our method by F1 score only on the smallest datasets, but the advantage is no longer on large datasets. We believe that when the F1 Score is comparable, the method with a higher Recall score is more advantageous. When the dataset is expanded from D_1 to D_3, the advantage of our method's Recall score over GraphSAGE expands from less than one percentage point to more than 8 percentage points. This indicates that PDTGA has better detection performance than other methods on large graphs, and can better detect phishing nodes on large-scale transaction networks by increasing the training samples and fully mining the information of Ethereum transactions between transaction nodes.

(2) The node representation capability of the PDTGA method increases with the enlargement of the transaction graph. During this experiment, we found that, as the number of nodes in the dataset increases from 29,539 to 39,854, the AUC score of PDTGA improves by 2.27%, the Recall score improves by 3.63%, and the classification performance advantage over other comparison methods expands. Taking the TTAGN method with the best performance in terms of AUC score as an example, the gap between its

Table 3
Feature set description.

Feature set name	Included feature categories
F	Temporal feature, node feature, edge feature
F\T	Node feature, edge feature
F\N	Temporal feature, edge feature
F\E	Temporal feature, node feature



(a) AUC (b) Recall

Fig. 6. AUC and Recall results of different kinds of feature ablation experiments.

Recall score and our method has widened from 8.47% to 14.61%. This shows that our PDTGA method can more effectively mine information that is difficult for other comparison methods to mine to help distinguish phishing accounts from normal accounts, while ignoring useless information, so with the expansion of the transaction graph, this advantage accumulates more obvious.

(3) Compared to all other methods, the feature-only method has the worst performance on the three datasets. In addition, its classification ability is degraded as the data expands, which may be caused by the lack of mining of topology information between nodes and information of neighbor nodes. However, it still has some classification ability, indicating that feature engineering of node features is still effective.

(4) The overall performance of the random walk method is significantly improved compared to the feature-only method. The LINE method performs the best, and the AUC score is the highest among the random walk methods. Compared with Node2Vec, the performance of the DeepWalk method decreases with the expansion of the dataset. This may be due to the fact that the DeepWalk sampling process is just a simple random walk compared to Node2Vec. The Node2Vec method is more conducive to expanding the depth and breadth of neighbor information in the search. In addition, methods such as random walk still ignore the transaction information between accounts, so the node representation is incomplete. This may be the reason why the classification performance of the LINE method is still slightly inferior to our method.

(5) The classification performance of the methods based on graph neural networks is close to our method. Such methods usually extract node features effectively and integrate them well into the spatial structure. This is also in line with the fact that phishing accounts can hide in a complex and practical network of transactions. The comprehensive performance of TTAGN is the second, only next to PDTGA, which is likely due to its ability of not only utilizing structural information between nodes like other conventional graph neural networks, but also of leveraging temporal information through an LSTM module. The performance of GAT is better than that of GraphSAGE and GCN, because GAT not only explores the label distribution when sampling, but also can effectively gather the features of neighbor nodes. Our method is improved from GAT, replacing the positional encoding module of GAT with a time encoding module, so that it has the ability to deal with dynamic graphs. In the embedding process, not only the features of neighbor nodes are aggregated, but also the features of edges related to the target node are aggregated. PDTGA enriches the characteristics of nodes, and also has the strongest representation ability of nodes in theory, of course, it is the same in practice.

5.4. Ablation study (RQ2)

5.4.1. Evaluate the validity of statistical features

To evaluate the contribution of three categories of statistical features (i.e., transaction time features, node features, and edge features) in the PDTGA detection model on four datasets (full feature set and three feature subsets), we conducted the feature ablation experiment by investigating the feature sets. The investigated feature sets are summarized in Table 3.

The data set D_1, D_2, D_3 are selected for the full feature sets. We remove the timestamp, node features and edge features of the full feature sets into three feature subsets respectively. We focus on examining the impact of three types of feature deletions on AUC and Recall scores, and the corresponding results are shown in Fig. 6.

As can be seen from Fig. 6, we can draw the following conclusions.

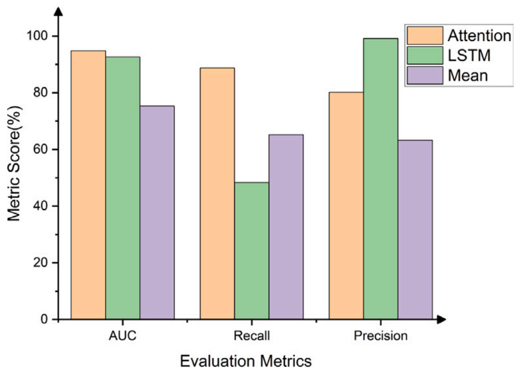


Fig. 7. Classification performance of different aggregation methods on dataset D_3.

(1) The PDTGA method has the best classification performance on the full feature set, and the worst performance when using the $F \backslash T$ feature subset. The transaction temporal feature is of great significance for improving the model's ability to detect phishing scam accounts. It also shows that the time encoding module can fully extract the temporal patterns of transaction interactions between nodes and learn more expressive node representations.

(2) The performance difference between using $F \backslash E$ and using the full set F is the smallest, which indicates that both edge features and node features have certain contributions to the detection of phishing accounts, but the correlation is strong. The possible reason for the analysis is that the account node features are based on statistics and calculations of all transaction data (i.e., the edge features) that the account participates in, so they share some hidden features. In Ethereum, the identity cannot pass real-name authentication, and there is no user profile information, so there is no portrait information of the node itself. The features of account nodes are calculated based on statistical features of amount, transaction direction, timestamps of account transactions, and the topology structure between accounts. Naturally, edge features are hidden in node features. Therefore, edge features contribute the least to the detection of phishing accounts.

(3) PDTGA performs better on all three full feature sets (i.e., $F \backslash T$, $F \backslash N$ and $F \backslash E$) than on their respective feature subsets, indicating that all three different types of features are helpful for node classification. At the same time, this advantage increases with the enlargement of the transaction graph, further indicating that PDTGA has an advantage in handling large-scale transaction graphs.

5.4.2. Evaluating the validity of multi-head self-attention aggregation

The TGAT model uses a multi-head self-attention mechanism, the essence of which is a linear transformation after the concatenation of multiple attention calculation results. This mechanism enables the model to use different feature information obtained at different positions, thereby increasing the diversity of features. In order to verify the effectiveness of multi-head self-attention aggregation, we change the aggregation method to mean aggregation and LSTM aggregation. To verify the effectiveness of multi-head self-attention aggregation, we designed a comparative experiment of changing the aggregation method to mean aggregation and LSTM aggregation. The specific results of the experiment are shown in Fig. 7.

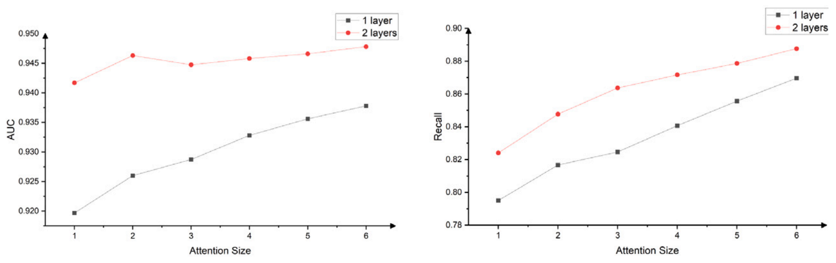
The experimental results show that although the LSTM aggregation method has achieved an AUC score close to that of the attention aggregation method, the Recall score is lower than 50%, which is obviously far from satisfying the actual deployment requirements. The overall performance of the mean aggregation method is even worse than the LSTM aggregation method.

5.5. Sensitivity analysis(RQ3)

Since the TGAT layer is a key component of node embeddings in the model, we explore the effect of its parameters on detection performance. The main parameters discussed here are the number of attention heads, the number of layers of convolution, the size of the embedding, and the learning rate. Sensitivity analysis experiments are all performed on dataset D_3.

5.5.1. Effect of attention sizes and layer numbers

First, we perform a sensitivity analysis on the attention heads and the number of layers of the TGAT embedding layer. We choose the number of TGAT layers from {1,2} and the number of attention heads from {1,2,3,4,5,6}. The TGAT embedding layer described in Section 4.5 stacks two layers by default to aggregate information from two-hop neighborhoods. We compare the classification performance of the model's embedding layer under different attention heads when using only a single layer and stacking two layers. The specific results are shown in Fig. 8. On the one hand, we can see that when stacking two layers, the AUC and Recall scores under different attention heads are higher than those of a single layer. It was concluded that stacking two layers resulted in a significant improvement compared to using only a single layer. On the other hand, as the number of attention heads increases, the AUC and Recall scores both increase when using only a single layer or when stacking two layers. It can be seen that increasing the number of attention heads helps to improve the model performance.



(a) AUC (b) Recall

Fig. 8. AUC and Recall results of different numbers of attention sizes and layers.

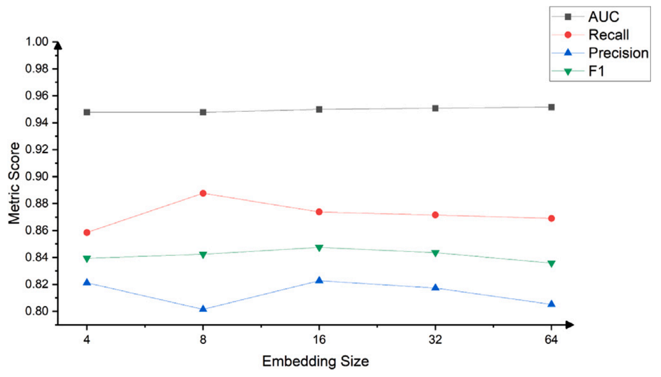


Fig. 9. Effect of embedding size.

5.5.2. Effect of embedding size

In this section, we investigate how the dimension of the embedding vector affects the representational ability of graph embeddings. Too high a vector dimension is easy to overfit, which will greatly increase the amount of calculation, bring about information redundancy, and on the contrary, reduce the representation ability. Conversely, too low vector dimension will also reduce the representation performance.

As can be seen from the results in Fig. 9, as the embedding dimension increases from 4 to 64, the AUC score increases by a small margin, the Recall and Precision fluctuate greatly, but the F1 score as a comprehensive evaluation of the Precision and Recall scores shows a flat trend of rising first and then falling. Taking these metrics into consideration, we believe that the best performance is when the embedding dimension is 16.

5.5.3. Effect of learning rate

We choose the initial learning rate from {0.001, 0.005, 0.0075, 0.01, 0.0125, 0.015, 0.02}. As can be seen from the results in Fig. 10, the AUC score of PDTGA shows a placid decreasing trend under different initial learning rates. It can be found that the AUC score of PDTGA shows a gentle decreasing trend when the learning rate increases from 0.001 to 0.02. While the Recall score first increased and then decreased, the Precision score first decreased and then increased. The comprehensive evaluation of the two, i.e., F1 score, first increased and then decreased, with a peak at 0.01. Therefore, we believe that the classification performance of the model with an initial learning rate of 0.01 is the best.

5.6. Summary

To answer RQ1, we evaluate the performance of the PDTGA method and all the compared methods on the Ethereum phishing scams detection task, and the experimental results show that our PDTGA method outperforms all other compared methods across the datasets. The results verify that temporal transaction information is of great significance to improve the ability of the model to detect phishing accounts in Ethereum. Compared with the limitations of existing methods, which lack temporal transaction information and weak node representation, PDTGA can effectively aggregate the dynamic node features of the neighbor nodes of the target node, the dynamic edge features of transactions related to the target node, and the topology of the target node between nodes interact.

We conduct feature ablation experiments to investigate RQ2. Experimental results show that when the dataset has no temporal transaction information, the classification performance is the worst. This demonstrates the contribution of temporal transaction information to classification. Both edge features and node features have a certain contribution to the detection of phishing scams.

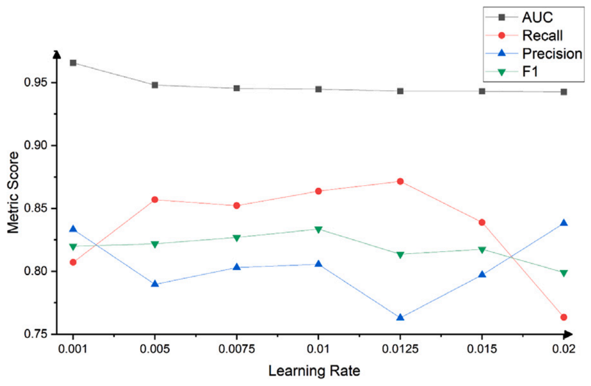


Fig. 10. Effect of learning rate.

But since edge features are partially hidden in node features, the contribution of edge features is smaller. This also verifies our idea that the introduction of time information and more complete node representation will help to improve the model's effectiveness in detecting phishing accounts in Ethereum. In addition, we also conduct ablation experiments on the attention aggregation method, which proves that attention aggregation can more efficiently aggregate more important information. That also contributes to phishing scams detection in Ethereum.

Finally, we performed sensitivity analysis experiments on the hyperparameters of the TGAT layer to answer RQ3. From the results, We found that a 2-layer TGAT layer with an attention size of 6 performed the best, the best embedding dimension was 16, and the learning rate was 0.01. It takes into account the precision and training speed.

6. Conclusion

In this work, we propose a method for applying graph representation learning based on temporal graph attention to improve the effectiveness of detecting phishing scams in Ethereum. We conducted experiments on three different sizes of datasets and benchmarked several models on this basis and found that our approach outperforms other baselines. It adapts better to large-scale datasets. In this paper, we also compare the contribution of different types of features to classification. The results show that the temporal features contribute the most, the edge features and node features both contribute to phishing detection, but edge features contribute less because they are partially hidden in node features. In addition, we find that attention aggregation can aggregate important information more effectively than LSTM aggregation and mean aggregation. This also helps in the detection of phishing scams in Ethereum. We also test the effect of different learning rates and embedding sizes on model performance and find that appropriately increasing the number of attention heads can improve the model performance.

Although PDTGA has achieved satisfactory results in Ethereum phishing scams detection, there are also some future directions to be investigated. First, this paper does not specifically consider the imbalance of the number of labeled and unlabeled nodes. We will study advanced oversampling as well as transformer approaches to enhance the effectiveness of PDTGA. Second, there are also differences in the behavior patterns of different phishers. This will make the transaction attributes of different edges on the transaction graph be different. We consider using a heterogeneous graph convolution model that supports different types of nodes and edges to further improve the effectiveness of PDTGA in the future.

CRediT authorship contribution statement

Lei Wang: Conceptualization, Methodology, Writing – original draft, Writing – review & editing. **Ming Xu:** Data curation, Software, Writing – original draft. **Hao Cheng:** Writing – original draft.

Data availability

Data will be made available on request.

Acknowledgments

This work is partially supported by the Six Talent Peaks project in Jiangsu Province, China (No. RJFW-029), the Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies (No. 2022B1212010005), and the Innovative training program for College Students in Jiangsu Province (No. 202110298077Y).

References

Abdelhamid, N., Ayesh, A., & Thabtah, F. (2014). Phishing detection based associative classification data mining. *Expert Systems with Applications*, *41*(13), 5948–5959.

ao Huang, Z., Sang, Y., Sun, Y., & Lv, J. (2022). A neural network learning algorithm for highly imbalanced data classification. *Information Sciences*, *612*, 496–513.

Bian, L., Zhang, L., Zhao, K., & Shi, F. (2020). Ethereum malicious account detection method based on LightGBM. *Netinfo Security*, *20*, 73–80.

Chen, W., Guo, X., Chen, Z., Zheng, Z., & Lu, Y. (2020). Phishing scam detection on ethereum: Towards financial security for blockchain ecosystem. In *IJCAI* (pp. 4506–4512).

Chen, H., Pendleton, M., Njilla, L., & Xu, S. (2020). A survey on ethereum systems security: Vulnerabilities, attacks, and defenses. *ACM Computing Surveys*, *53*, 43.

Chen, L., Peng, J., Liu, Y., Li, J., Xie, F., & Zheng, Z. (2021). Phishing scams detection in ethereum transaction network. *ACM Transactions on Internet Technology (TOIT)*, *21*, 16.

Dong, Y., Cordonnier, J.-B., & Loukas, A. (2021). Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International conference on machine learning* (pp. 2793–2803). PMLR.

Estevam, G., Palma, L. M., Silva, L. R., Martina, J. E., & Vigil, M. (2021). Accurate and decentralized timestamping using smart contracts on the ethereum blockchain. *Information Processing & Management*, *58*(3), Article 102471.

Fan, S., Fu, S., Xu, H., & Cheng, X. (2021). Al-SPSD: Anti-leakage smart Ponzi schemes detection in blockchain. *Information Processing & Management*, *58*(4), Article 102587.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *International conference on machine learning* (pp. 1263–1272).

Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 855–864).

Hamilton, W. L., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in neural information processing systems, Vol. 30* (pp. 1024–1034).

Haruta, S., Asahina, H., & Sasase, I. (2018). Visual similarity-based phishing detection scheme using image and css with target website finder. In *Globecom IEEE global communications conference* (pp. 1–6).

Hu, T., Liu, X., Chen, T., Zhang, X., Huang, X., Niu, W., et al. (2021). Transaction-based classification and detection approach for ethereum smart contract. *Information Processing & Management*, *58*(2), Article 102462.

Huang, H., Kong, W., Zhou, S., Zheng, Z., & Guo, S. (2021). A survey of state-of-the-art on blockchains: Theories, modelings, and tools. *ACM Computing Surveys*, *54*(2), 44:1–44:42.

Jourdan, M., Blandin, S., Wynter, L., & Deshpande, P. (2018). Characterizing entities in the bitcoin blockchain. In *2018 IEEE international conference on data mining workshops* (pp. 55–62). IEEE.

Kumar, N., Singh, A., Handa, A., & Shukla, S. K. (2020). Detecting malicious accounts on the ethereum blockchain with supervised learning. In *International symposium on cyber security cryptography and machine learning* (pp. 94–109). Springer.

Leng, J., Zhou, M., Zhao, J. L., Huang, Y., & Bian, Y. (2022). Blockchain security: A survey of techniques and research directions. *IEEE Transactions on Services Computing*, *15*(4), 2490–2510.

Li, S., Gou, G., Liu, C., Hou, C., Li, Z., & Xiong, G. (2022). TTAGN: Temporal transaction aggregation graph network for ethereum phishing scams detection. In *Proceedings of the ACM web conference 2022* (pp. 661–669).

Li, Y., Yang, Z., Chen, X., Yuan, H., & Liu, W. (2019). A stacking model using URL and HTML features for phishing webpage detection. *Future Generation Computer Systems*, *94*, 27–39.

Li, L., Zhao, K., Gan, J., Cai, S., Liu, T., Mu, H., et al. (2021). Robust adaptive semi-supervised classification method based on dynamic graph and self-paced learning. *Information Processing & Management*, *58*(1), Article 102433.

Lin, D., Wu, J., Yuan, Q., & Zheng, Z. (2020). Modeling and understanding ethereum transaction records via a complex network approach. *IEEE Transactions on Circuits and Systems II: Express Briefs*, *67*(11), 2737–2741.

Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. (2019). Machine learning based phishing detection from URLs. *Expert Systems with Applications*, *117*, 345–357.

Sun, H., Ruan, N., & Liu, H. (2019). Ethereum analysis via node clustering. In *International conference on network and system security* (pp. 114–129). Springer.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web* (pp. 1067–1077).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 5998–6008.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. *Statistics*, *1050*, 20.

Wang, L., Cheng, H., Zheng, Z., Yang, A., & Zhu, X. (2021). Ponzi scheme detection via oversampling-based long short-term memory for smart contracts. *Knowledge-Based Systems*, *228*, Article 107312.

Wang, B., Liu, H., Liu, C., Yang, Z., Ren, Q., Zheng, H., et al. (2021). BLOCKEYE: Hunting for DeFi attacks on blockchain. In *2021 IEEE/ACM 43rd international conference on software engineering: companion proceedings (ICSE-companion)* (pp. 17–20).

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., & Solomon, J. M. (2019). Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics*, *38*(5), 146:1–146:12.

Welling, M., & Kipf, T. N. (2016). Semi-supervised classification with graph convolutional networks. In *J. international conference on learning representations (ICLR 2017)* (pp. 1–14).

Wen, T., Xiao, Y., Wang, A., & Wang, H. (2023). A novel hybrid feature fusion model for detecting phishing scam on ethereum using deep neural network. *Expert Systems with Applications*, *211*, Article 118463.

Williams, E. J., & Polage, D. (2019). How persuasive is phishing email? The role of authentic design, influence and current events in email judgements. *Behaviour & Information Technology*, *38*(2), 184–197.

Wu, H., Gu, X., & Gu, Y. (2017). Balancing between over-weighting and under-weighting in supervised term weighting. *Information Processing & Management*, *53*(2), 547–557.

Wu, J., Yuan, Q., Lin, D., You, W., Chen, W., Chen, C., et al. (2020). Who are the phishers? Phishing scam detection on ethereum via network embedding. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *52*(2), 1156–1166.

Xie, Y., Zhou, J., Wang, J., Zhang, J., Sheng, Y., Wu, J., et al. (2021). Understanding ethereum transactions via network approach. In *Graph data mining* (pp. 155–176). Springer.

Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., & Achan, K. (2019). Self-attention with functional time representation learning. In *Advances in neural information processing systems, Vol. 32* (pp. 15889–15899).

Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., & Achan, K. (2020). Inductive representation learning on temporal graphs. In *Proc. ICLR* (pp. 1–19).

Yin, H. S., & Vatrapu, R. (2017). A first estimation of the proportion of cybercriminal entities in the bitcoin ecosystem using supervised machine learning. In *2017 IEEE international conference on big data (big data)* (pp. 3690–3699). IEEE.

Yu, T., Chen, X., Xu, Z., & Xu, J. (2022). MP-GCN: A phishing nodes detection approach via graph convolution network for ethereum. *Applied Sciences*, *12*(14), 7294.

Zeng, L., Li, H., Xiao, T., Shen, F., & Zhong, Z. (2022). Graph convolutional network with sample and feature weights for Alzheimer's disease diagnosis. *Information Processing & Management*, *59*(4), Article 102952.

Zhao, Y., Xie, Y., Yu, F., Ke, Q., Yu, Y., Chen, Y., et al. (2009). BotGraph: Large scale spamming botnet detection. In *Proceedings of the 6th USENIX symposium on networked systems design and implementation* (pp. 321–334).

Zheng, Z., Xie, S., Dai, H.-N., Chen, X., & Wang, H. (2018). Blockchain challenges and opportunities: a survey. *International Journal of Web and Grid Services*, *14*(4), 352–375.