Information Fusion 120 (2025) 103074

Contents lists available at ScienceDirect

Information Fusion

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

Full length article

Ethereum fraud detection via joint transaction language model and graph representation learning

Jianguo Sun [a,c], Yifan Jia [a], Yanbin Wang [b,c,*], Ye Tian [c], Sheng Zhang [d]

[a] Yantai Research Institute, Harbin Engineering University, Yantai, Shandong, China
[b] Department of Engineering, Shenzhen MSU-BIT University, Shenzhen, China
[c] Hangzhou Research Institute, Xidian University, Hangzhou, Zhejiang, China
[d] Hefei University of Technology, China

ARTICLE INFO

Dataset link: https://github.com/lincozz/TLmGNN

Keywords:
Ethereum fraud detection
Transaction language model
Graph information fusion
Joint training

ABSTRACT

Ethereum faces growing fraud threats. Current fraud detection methods, whether employing graph neural networks or sequence models, fail to consider the semantic information and similarity patterns within transactions. Moreover, these approaches do not leverage the potential synergistic benefits of combining both types of models. To address these challenges, we propose TLMG4Eth that combines a transaction language model with graph-based methods to capture semantic, similarity, and structural features of transaction data in Ethereum. We first propose a transaction language model that converts numerical transaction data into meaningful transaction sentences, enabling the model to learn explicit transaction semantics. Then, we propose a transaction attribute similarity graph to learn transaction similarity information, enabling us to capture intuitive insights into transaction anomalies. Additionally, we construct an account interaction graph to capture the structural information of the account transaction network. We employ a deep Multi-Head Attention Network to fuse transaction semantic and similarity embeddings, and ultimately propose a joint training approach for the Multi-Head Attention Network and the account interaction graph to obtain the synergistic benefits of both. Our model achieves performance improvements ranging from 9.62% to 13.2% over state-of-the-art methods on two public datasets and a newly introduced dataset. Our code is available at the following link: https://github.com/lincozz/TLmGNN.

1. Introduction

Blockchain technology has revolutionized various industries by providing a decentralized and secure method for recording transactions [1]. Among blockchain platforms, Ethereum stands out for its robust support of smart contracts and decentralized applications [2]. By introducing the concept of a programmable blockchain, Ethereum enabled developers to create applications beyond basic financial transactions [3]. This innovation has established Ethereum as a foundational layer for numerous blockchain applications, including decentralized finance (DeFi) and non-fungible tokens (NFTs) [4–7].

The growing adoption and value of Ethereum have, however, attracted malicious actors intent on exploiting the platform for financial gain. Fraudulent activities within the Ethereum ecosystem primarily include phishing, Ponzi schemes, transaction manipulation, and counterfeit dApps. Alarmingly, phishing scams constitute a significant portion of malicious fraud within the blockchain ecosystem, accounting for approximately 50% of such incidents [8]. Phishing and fraud have become significant challenges within the Ethereum ecosystem. Phishing typically involves deceiving users into revealing sensitive information for financial gain, often via fake websites or messages [9]. Fraud spans a range of activities, including transaction manipulation and counterfeit dApps, aimed at deceiving users or systems for profit. The scale of these threats is evident in recent data. According to the Chainalysis 2023 Crypto Crime Report [10], USD 39.6 billion worth of crypto-assets were received by identified illicit addresses, representing 0.42% of total on-chain transaction volume—an increase from the previous year's USD 23.2 billion. This trend highlights the urgent need for effective detection and mitigation measures within the Ethereum network.

Ethereum fraud detection primarily relies on analyzing historical transaction records of accounts. Current methods predominantly employ Graph Neural Networks (GNNs) to model account transaction networks, or utilize sequence models such as Transformers [11] to process transaction histories. GNNs excel at capturing complex relationships and structural patterns within transaction networks [12–14],

* Corresponding author.
E-mail addresses: jgsun@xidian.edu.cn (J. Sun), wangyanbin15@mails.ucas.ac.cn (Y. Wang).

https://doi.org/10.1016/j.inffus.2025.103074
Received 25 November 2024; Received in revised form 24 February 2025; Accepted 1 March 2025
Available online 13 March 2025

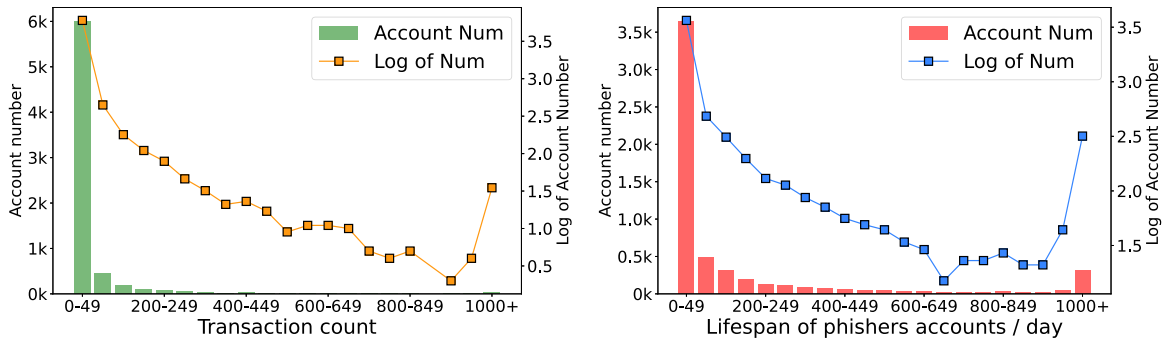

Fig. 1. The distribution patterns of the transaction count and lifespan of identified phishing accounts.

while sequence models are proficient in discerning temporal patterns and evolving behaviors [15–17].

However, current research fails to consider several crucial aspects: (1) Transaction Semantics: Existing approaches rely on historical transaction data in its numerical form, which lacks the context to interpret underlying intentions. As a result, the explicit meaning of transactions remains obscure, hindering models from understanding transaction semantics beyond mere numbers. (2) Transactional Similarity: Extracting similarity information from transaction attributes (such as amount, direction, and timing) is crucial for distinguishing between normal and anomalous transactions. Previous studies have overlooked the modeling of attribute similarities, which can offer direct insights into fraudulent behavior patterns. (3) Synergistic Optimization: While some studies have attempted to combine GNNs with sequence models, they typically adopt a late fusion approach, training the models separately and concatenating their features at the final stage. This approach fails to fully realize the potential synergies between the two methods.

To address the current challenges, we propose TLMG4Eth, which combines a transaction language model (TLM) with two transaction graphs to enhance Ethereum fraud detection, offering new insights for strengthening the security of the decentralized finance ecosystem. We first train a transaction language model to learn explicit transaction semantics from transaction sentences, where transaction attributes (amount, direction, time, and other numerical data) are represented as words. Next, we propose a transaction attribute graph to model global semantic similarities between transactions and construct an account interaction graph to model transaction behaviors between accounts. We fuse transaction semantics, similarities, and structural information through a two-stage approach. First, we use a deep Multi-Head Attention Network to fuse the semantic embeddings and similarity embeddings of transactions. Then, we propose to jointly train the Multi-Head Attention Network with the account interaction graph to leverage their synergistic benefits.

Our main contributions include:

- We propose a transaction language model that transforms numerical transaction sequences into transaction sentences, clearly expressing transaction content and enabling the learning of explicit transaction semantics.
- We propose a transaction attribute similarity graph to model global semantic similarities between transactions, thereby capturing intuitive insights into transaction anomalies.
- We use a Multi-Head Attention Network to fuse transaction semantic and similarity information. Furthermore, we propose jointly training this Multi-Head Attention Network with an account interaction graph to obtain the benefits of both.
- Our proposed method significantly outperforms current state-of-the-art approaches, improving F1 Scores by 9.62%%-13.2% across three datasets.
- We release a new dataset providing an up-to-date view of phishing activities on the Ethereum transaction network, facilitating further research in this area.

2. Background and related work

2.1. Background

In the Ethereum network, there are two main types of accounts: Externally Owned Accounts (EOAs) and Contract Accounts.

Externally Owned Accounts (EOAs) EOAs are controlled by private keys held by users. These accounts can initiate transactions to transfer cryptocurrency or trigger contract executions. We primarily focus on EOAs because they are directly controlled by humans, making them more susceptible to phishing and other fraudulent activities.

Contract Accounts These accounts are essentially smart contracts, which are self-executing programs running on the Ethereum blockchain. Contract accounts cannot initiate transactions themselves but can execute internal transactions when triggered by EOAs [18,19].

Internal Transactions These transactions are initiated by smart contracts and occur within the blockchain. They are typically used for complex operations within contracts and are not directly initiated by users.

External Transactions These are transactions initiated by EOAs, involving the transfer of cryptocurrency to other EOAs or contract accounts. External transactions are our primary focus because they directly reflect user activities and are more likely to reveal fraudulent behavior [20,21].

The focus on EOAs and external transactions is crucial for detecting fraud because these transactions provide clear insights into user behavior and potential phishing activities. Internal transactions, while important, do not offer the same direct evidence of user-controlled activities.

Phishing accounts on Ethereum tend to have very short lifespans. This is because once a phishing account is identified and reported, it is quickly flagged and often deactivated by the community or relevant authorities. As shown in Fig. 1 our investigation into 7067 identified phishing accounts revealed that 6454 of these accounts have fewer than 100 transactions each. This indicates that phishing operations often involve creating multiple accounts that execute a limited number of transactions to avoid detection and maximize their impact before being flagged.

2.2. Related work

Graph-based methods Graph-based methods construct transaction networks between accounts and employ graph embedding algorithms or GNNs for model training. [22–24] use Node2Vec to extract features from Ethereum transaction networks for fraudulent account detection. Trans2Vec [25] utilizes DeepWalk [26], dividing representation learning into node, edge, and attribute learning for classification. TGC [8] employs subgraph contrastive learning with statistical data for phishing address identification. [27,28] construct statistical features based on transaction records and apply these features to the graph representation learning method based on the attention mechanism. SIEGE [29]

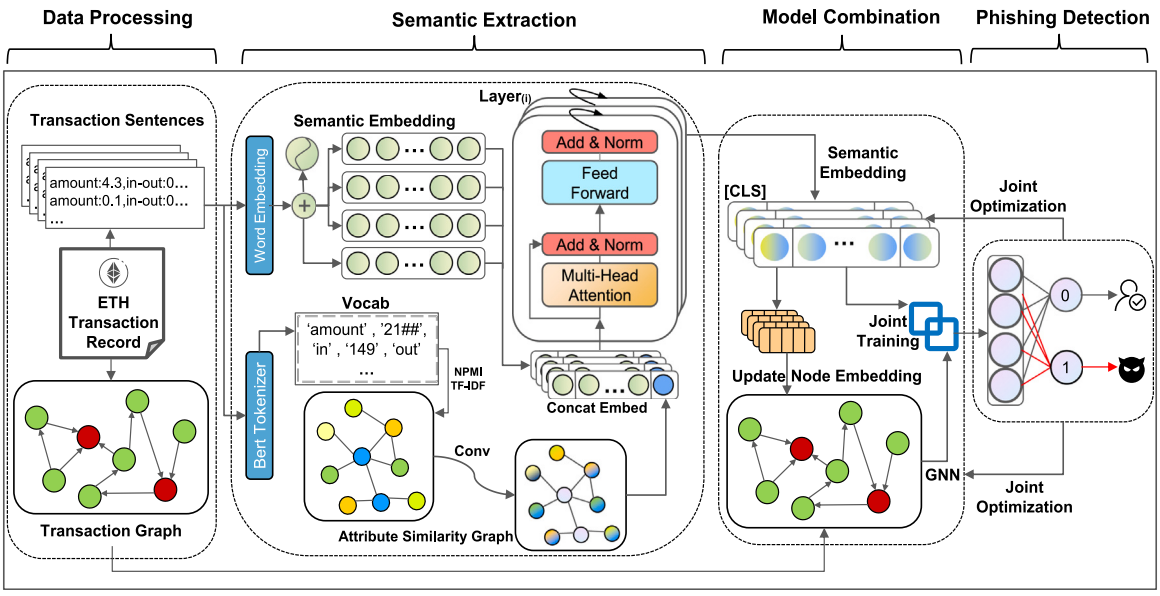

Fig. 2. The framework of proposed Joint Transaction Language Model and Graph Representation Learning.

proposes a self-supervised incremental graph learning model, which effectively improves the performance of Ethereum phishing fraud detection through spatiotemporal pre-tasks and incremental learning. TokenScout [30] uses time graph learning technology to build a dynamic graph model to monitor Ethereum token trading behavior and achieve early detection of fraudulent tokens.

Sequence-based methods Graph-based approaches may struggle with high-frequency, repetitive transactions and long-term temporal patterns, leading some researchers to adopt sequence models that treat transactions as time-ordered event streams. BERT4ETH [31] and ZipZap [32] exemplifies this approach, employing a BERT-like structure with a Transformer architecture to process chronological transaction events. It uses a masked language model for pre-training by randomly masking transaction addresses, then fine-tunes a Multi-Layer Perceptron (MLP) network for account classification.

Hybrid methods TSGN [33] introduces a transaction subgraph network model for phishing detection, integrating Handcrafted and Diffpool features while enhancing classification with diverse mapping mechanisms. Similarly, TTAGN [34] adopts a multi-step approach, leveraging Edge2Node for edge aggregation, time-based transaction graphs with LSTM for temporal patterns, and statistical feature extraction, ultimately combining these representations for classification. [35] proposes a new Ethereum transaction fraud detection technology based on a stacking method, which effectively improves the ability to identify fraudulent behavior by integrating the advantages of multiple models. [36] uses a method that combines convolutional neural networks and XGBoost classifiers to distinguish normal accounts from illegal accounts based on transaction history. Grabphisher [37] extracts account features by extracting account temporal features and capturing dynamic topological information during graph evolution, constructing the evolution pattern of transaction accounts into a continuous-time diffusion network graph.

Existing graph-based deep learning and graph embedding methods have made significant strides in fraud detection by effectively capturing structural properties and complex dependencies within transaction networks. However, they fail to preserve the transaction order, overlook position embeddings, and lack the integration of semantic information from transaction attributes with structural data. These limitations hinder their ability to fully understand transaction contexts. Although Transformer-based sequence models can capture transaction sequence information, they are less effective than graph-based methods in capturing key account interactions and topological information. Specifically, they still rely on context-independent numerical features of accounts and fail to incorporate explicit transaction semantics. Our proposed TLMG4Eth method addresses these challenges by combining a transaction language model with a multi-head attention network, effectively capturing both semantic and structural aspects to enhance detection performance.

Beyond individual model improvements, traditional model-ensemble approaches, while leveraging the strengths of multiple models, often suffer from limited interaction and inter-model communication. Typically, these methods treat each model as an isolated entity, merely aggregating outputs post-hoc without fostering dynamic collaboration during training. In contrast, our approach employs a joint training framework, enabling continuous feedback and shared parameter optimization among models. This synergistic interaction not only enhances individual model performance but also allows them to dynamically adapt to each other's strengths and weaknesses, resulting in a more cohesive and robust system capable of capturing complex patterns and interdependencies within transaction data.

3. Method

In this section, we introduce in detail how TLMG4Eth integrates the pre-trained language model with GNNs to achieve the fusion of sequence and network structure information. The architecture of TLMG4Eth is depicted in 2. For a clearer understanding, our explanation is divided into the following parts: Transaction Language Model, Transaction Attribute Similarity Graph, Semantic and Similarity Embedding Fusion, Account Interaction Graph, Joint Training of MAN and AIG. Several important notations used in this paper are summarized in Table 1.

3.1. Transaction language model

The Transaction Language Model (TLM) consists of two main parts: first, we create a linguistic representation of numerical transaction data; second, we employ a language model to extract semantic embeddings from transaction sentences.

3.1.1. Linguistic representation of transactions

Numerical transaction data often obscures specific transaction information. To address this, we propose a linguistic representation of transactions to elucidate their content.

Let $\mathcal{T} = \{t_1, t_2, \ldots, t_N\}$ be the set of N transactions associated with a single account. Each transaction t_i is characterized by a tuple:

$$t_i = (v_i, d_i, \tau_i) \tag{1}$$

where:

- v_i is the transaction amount.
- $d_i \in \{-1, 1\}$ is the transaction direction, with -1 indicating an inflow and 1 indicating an outflow.
- $\tau_i \in \mathbb{T}$ is the timestamp of the transaction, where $\mathbb{T}$ is the set of all possible timestamps.

We transform each numerical attribute into a linguistic token by prepending a descriptive text indicator:

$$\mathcal{L}(t_i) = \{\text{amount: } v_i,\ \text{direction: } d_i,\ \text{timestamp: } \tau_i\} \tag{2}$$

However, raw Ethereum timestamps (e.g., 2024121214) lack interpretability and may mislead models due to their large numerical values. To address this, we capture the intervals between consecutive transactions rather than using raw timestamps.

Let τ_i denote the timestamp of transaction t_i. We define the time intervals as:

$$\Delta\tau_{i,n} = \tau_i - \tau_{i-n}, \quad \text{for } n \in \{1, 2, 3, 4, 5\} \tag{3}$$

where $\Delta\tau_{i,n}$ represents the time difference between transaction t_i and its nth preceding transaction.

We incorporate the time differences from the 2nd to the 5th preceding transactions into $\mathcal{L}(t_i)$. The enhanced representation $\mathcal{L}'(t_i)$ is defined as:

$$\begin{aligned}\mathcal{L}'(t_i) = \{&\text{amount: } v_i,\ \text{direction: } d_i,\\ &\text{2-inter\_time: } \Delta\tau_{i,2},\\ &\text{3-inter\_time: } \Delta\tau_{i,3},\\ &\text{4-inter\_time: } \Delta\tau_{i,4},\\ &\text{5-inter\_time: } \Delta\tau_{i,5}\}\end{aligned} \tag{4}$$

This enhanced representation captures transaction clustering at different time granularities. Each element in $\mathcal{L}'(t_i)$ is treated as a transaction word.

The N transactions of an account form a series of transaction sentences C:

$$C = \{\mathcal{L}'(t_1), \mathcal{L}'(t_2), \ldots, \mathcal{L}'(t_N)\} \tag{5}$$

3.1.2. Transaction semantic embedding

We employ BERT-base [38] to extract semantic embeddings from these transaction representations. We continue training BERT using our domain-specific pre-training corpus, denoted as $\mathcal{D}$:

$$\mathcal{D} = \bigcup_{a \in \mathcal{A}} C_a \tag{6}$$

where $\mathcal{A}$ is the set of all accounts, and C_a is the transaction sentences for account a.

We use a masked language model (MLM) objective:

$$\mathcal{L}_{\text{MLM}} = \mathbb{E}_{t \sim \mathcal{D}}\left[-\sum_{i \in \mathcal{M}} \log P(x_i | \tilde{t})\right] \tag{7}$$

where $\mathcal{M}$ is the set of masked token indices, $\tilde{t}$ is the masked version of transaction sentence t, x_i is the ith token in t, and $P(x_i|\tilde{t})$ is the probability of predicting the original token x_i given the masked context.

After pre-training, for each token x_i in a transaction sentence t, BERT-base generates a semantic embedding vector:

$$\mathbf{e}_i^s = \text{BERT}(x_i | t) \in \mathbb{R}^d \tag{8}$$

where d is the dimensionality of the embedding space.

Table 1
Key notation and description.

Notations	Descriptions
$\mathcal{T}$	Set of N transactions for a single account
t_i	Transaction i, characterized by (v_i, d_i, τ_i)
v_i	Transaction amount in t_i
d_i	Transaction direction in t_i; -1 inflow, 1 outflow
τ_i	Timestamp of transaction t_i
$\Delta\tau_{i,n}$	Time difference of t_i and t_{i-n}
$\mathcal{L}(t_i)$	Linguistic representation of transaction t_i
$\mathcal{L}'(t_i)$	Enhanced linguistic representation of transaction t_i
C	Sequence of transaction sentences for an account
$\mathcal{A}$	Set of all accounts
$\mathcal{D}$	Corpus of all transaction sentences
$\mathcal{G}_w = (\mathcal{V}_w, \mathcal{E}_w)$	TASG with word nodes and edges
w_i, w_j	Words in the vocabulary
$\text{NPMI}(w_i, w_j)$	NPMI between w_i and w_j
$\text{TF-IDF}(w_i, d)$	TF-IDF score of word w_i in sentence d
θ	Predefined threshold for NPMI or TF-IDF
$\mathbf{e}_i^g$	Similarity embedding of word w_i from TASG
$\mathbf{E}_i$	Concatenated embedding $[\mathbf{e}_i^s; \mathbf{e}_i^g]$
$\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$	Query, key, and value matrices in attention
d_k	Dimensionality of key vectors
$\mathbf{Z}$	Output from Multi-Head Attention Network (MAN)
$\mathbf{H}$	Feature vector corresponding to the [CLS] token
$\mathcal{G}_a = (V_a, E_a)$	Account Interaction Graph
V_a	Set of account nodes in AIG
E_a	Set of edges between accounts in AIG
w_{ij}	Weight of edge between accounts i and j
$\mathbf{A}$	Adjacency matrix of AIG
$\mathbf{X}$	Initial node features for GCN from MAN outputs
$\mathbf{H}^{(l)}$	Node features at layer l of GCN
$\hat{\mathbf{A}}$	Normalized adjacency matrix with self-loops
$\sigma(\cdot)$	Activation function
$\mathbf{Z}_{\text{GCN}}$	Output of GCN after processing node embeddings
$\mathbf{Z}_{\text{MAN}}$	Prediction from MAN
λ	Hyperparameter balancing MAN and GCN outputs
Pred	Final prediction after combining MAN and GCN
y_i	Ground-truth label for sample i
$\mathcal{L}$	Cross-entropy loss function

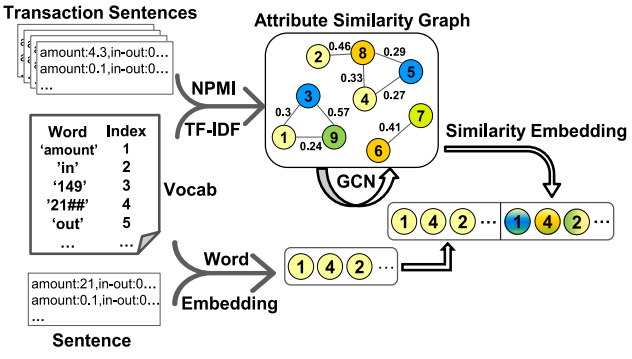

Fig. 3. The generation and combination of ethereum transaction semantic embedding and similarity embedding.

3.2. Transaction attribute similarity graph

While pretrained language models can capture semantic features from account transaction records, their embeddings rely solely on individual account histories and lack sensitivity to anomalous terms within transaction texts—critical information for distinguishing phishing accounts. For instance, phishing accounts often execute large transactions within short intervals, exhibiting similar semantic patterns in both transaction amounts and temporal behavior. To address this limitation, we propose the Transaction Attribute Similarity Graph (TASG), which captures global transaction correlations across Ethereum data, offering intuitive insights into anomalies.

We generate a vocabulary from the tokenized transaction corpus. Subsequently, we construct the TASG using two approaches: Normalized Pointwise Mutual Information (NPMI) and Term Frequency-Inverse Document Frequency (TF-IDF). Let TASG be denoted as $\mathcal{G}_w = (\mathcal{V}_w, \mathcal{E}_w)$, where $\mathcal{V}_w$ represents the set of nodes corresponding to words in the vocabulary, and $\mathcal{E}_w$ represents the set of edges connecting these nodes. The presence of an edge between two nodes is determined by their NPMI or TF-IDF value.

The NPMI [39] between two words w_i and w_j is calculated as follows:

$$\text{NPMI}(w_i, w_j) = \frac{\log \frac{p(w_i, w_j)}{p(w_i)p(w_j)}}{-\log p(w_i, w_j)} \tag{9}$$

where $p(w_i, w_j)$ is the probability of co-occurrence of w_i and w_j within a given context window, and $p(w_i)$ and $p(w_j)$ are the individual probabilities of w_i and w_j. In our approach, the window size is the length of one transaction sentence, and we create an edge between two words if their NPMI exceeds a predefined threshold θ.

The TF-IDF [40] score of word w_i in transaction sentence d is calculated as follows:

$$\text{TF-IDF}(w_i, d) = \text{TF}(w_i, d) \times \log\left(\frac{N}{|\{d' \in D : w_i \in d'\}|}\right) \tag{10}$$

where $\text{TF}(w_i, d)$ is the term frequency of word w_i in transaction sentence d, N is the total number of sentences in the corpus D, and $|\{d' \in D : w_i \in d'\}|$ is the number of sentences containing w_i. In our approach, we introduce additional sentence nodes in the vocabulary graph to model a TF-IDF-based TASG $\mathcal{G}_w$; the connectivity between sentence nodes and word nodes depends on whether their TF-IDF values exceed the predefined threshold θ.

We then apply a Graph Convolutional Network (GCN) to encode nodes in the TASG, obtaining global similarity embeddings $\mathbf{e}_i^g$ for each word w_i. Intuitively, NPMI can capture tokens with high co-occurrence frequency in transaction sentences, while TF-IDF can reveal the token in the transaction sentence that can best identify an account. These two strategies provide fine-grained information supplements for clarifying the semantics of account transactions from the perspectives of word similarity and discriminability, respectively, alleviating the deficiency of the semantic embedding obtained by language modeling that lacks global information.

3.3. Semantic and similarity embedding fusion

Since the vocabulary used for generating the transaction sequence semantic embeddings and constructing the vocabulary graph are both derived from the same tokenizer, the words in each account's transaction sequence are a subset of the vocabulary graph [41]. As shown in Fig. 3, we select the corresponding word nodes from the TASG based on the input transaction sequence and concatenate the transaction similarity embeddings generated by TASG with the semantic embeddings generated by TLM.

For each token x_i in the transaction sequence, we obtain its semantic embedding $\mathbf{e}_i^s$ from the TLM and its similarity embedding $\mathbf{e}_i^g$ from the TASG GCN encoder. We then concatenate them:

$$\mathbf{E}_i = [\mathbf{e}_i^s; \mathbf{e}_i^g] \tag{11}$$

where $[\cdot;\cdot]$ denotes concatenation along the feature dimension.

We fuse the information from the two types of embeddings $\mathbf{E}_i$ using a deep Multi-Head Attention Network (MAN) that consists of 12 layers, each with 12 attention heads. The computation for each attention head is as follows:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \tag{12}$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are the query, key, and value matrices, respectively, and d_k is the dimensionality of the key vectors.

The multi-head attention involves several independent attention heads, each computing the scaled dot-product attention. Through these attention mechanisms, in a 12-layer, 12-head self-attention encoder, semantic embeddings and similarity embeddings interact layer by layer to achieve fusion. Each word vector obtains a representation encoding contextual information. The corresponding output is denoted as:

$$\mathbf{Z} = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)\mathbf{W}_O \tag{13}$$

where $\mathbf{Z} \in \mathbb{R}^{B\times L\times F}$, B is the batch size, L is the sequence length, F is the dimension of the multi-head attention hidden layer, h is the number of attention heads (here $h = 12$), and $\mathbf{W}_O$ is the learned projection matrix that linearly transforms the concatenated outputs from all attention heads into the final dimensionality required by the model.

By extracting the hidden state $\mathbf{H} \in \mathbb{R}^{B\times F}$ corresponding to the "[CLS]" token (the first token) from the final output of the encoder, we obtain the global feature vector $\mathbf{H}$ of the sequence and get the final representation of the transaction sequence by the transaction language model. This representation is applied to subsequent classification tasks.

3.4. Account interaction graph

We construct an Account Interaction Graph (AIG) to model transaction relationships between accounts and capture the topological information of transactions. We represent the AIG as a weighted graph $\mathcal{G}_a = (V_a, E_a)$, where V_a is the set of account nodes in the transaction network, with each node representing an individual account, and E_a is the set of edges, where each edge (i, j) represents that account i has transacted with account j. The weight of an edge w_{ij} represents the number of transactions between account i and account j.

The adjacency matrix of the graph is denoted as $\mathbf{A} \in \mathbb{R}^{N_a\times N_a}$, where $N_a = |V_a|$ is the number of account nodes. The weight matrix $\mathbf{W} \in \mathbb{R}^{N_a\times N_a}$ contains the transaction counts between pairs of nodes [42].

We initialize the account node features $\mathbf{X} \in \mathbb{R}^{N_a\times F}$ using the transaction sequence representations obtained from the TLM in the previous section, where F is the feature dimension.

The account nodes are iteratively updated using a GCN. The update rule for layer $l+1$ is given by:

$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right) \tag{14}$$

where:

- $\mathbf{H}^{(l)} \in \mathbb{R}^{N_a\times F^{(l)}}$ is the node feature matrix at layer l.
- $\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I})\mathbf{D}^{-\frac{1}{2}}$ is the normalized adjacency matrix with self-loops added ($\mathbf{I}$ is the identity matrix and $\mathbf{D}$ is the degree matrix).
- $\mathbf{W}^{(l)} \in \mathbb{R}^{F^{(l)}\times F^{(l+1)}}$ is the trainable weight matrix for layer l.
- $\sigma(\cdot)$ is a non-linear activation function, such as ReLU.

3.5. Joint training of MAN and AIG

We combine the Multi-Head Attention Network (MAN) and the Account Interaction Graph (AIG) for joint model training.

First, we use the output $\mathbf{H}$ from the MAN as the initial node embeddings $\mathbf{X}$ for the GCN on the AIG, The GCN processes $\mathbf{X}$ and outputs node representations $\mathbf{Z}_{\text{GCN}}$:

$$\mathbf{X} = \mathbf{H} \tag{15}$$

$$\mathbf{Z}_{\text{GCN}} = \text{GCN}(\mathbf{X}, \mathbf{A}) \tag{16}$$

We also obtain the prediction $\mathbf{Z}_{\text{MAN}}$ from the MAN by applying a classification layer on $\mathbf{H}$:

$$\mathbf{Z}_{\text{MAN}} = \text{softmax}(\mathbf{W}_{\text{MAN}}\mathbf{H} + \mathbf{b}_{\text{MAN}}) \tag{17}$$

where $\mathbf{W}_{\text{MAN}}$ and $\mathbf{b}_{\text{MAN}}$ are the weights and biases of the classification layer.

Then, we linearly interpolate the predictions from the MAN and GCN to obtain the final prediction:

$$\text{Pred} = \lambda \mathbf{Z}_{\text{GCN}} + (1-\lambda)\mathbf{Z}_{\text{MAN}} \tag{18}$$

where $\lambda \in [0,1]$ is a hyperparameter controlling the balance between the two models' contributions to the result. $\lambda = 0$ means that we use only the MAN model, while $\lambda = 1$ means that we use only the GCN model.

Algorithm 1: Process of TLMG4Eth Framework

Input: Account set - $\mathcal{A}$
Transaction data - $\mathcal{T}$
Labels - $\mathcal{Y}$
Learning rate - η
Batch size - B
Threshold for TASG - θ
Interpolation weight - λ
Number of epochs - E
Output: Trained model parameters Θ
Function TLMG4Eth():
  ▷ Initialize model parameters
  $\Theta \leftarrow \{\Theta_{\text{TLM}}, \Theta_{\text{TASG}}, \Theta_{\text{MAN}}, \Theta_{\text{AIG}}, \Theta_{\text{MLP}}\}$
  $\mathcal{V} \leftarrow \text{BuildVocabulary}(\mathcal{T})$
  $\mathcal{G}_w \leftarrow \text{ConstructTASG}(\mathcal{V}, \theta)$
  for *epoch* $= 1$ **to** E **do**
    for $\{(a_i, y_i)\}_{i=1}^{B} \in \textit{MiniBatches}(\mathcal{A}, \mathcal{Y})$ **do**
      ▷ Extract and encode transaction features
      for $a_i \in \mathcal{A}$ **do**
        $\mathcal{T}_{a_i} \leftarrow \text{GetTransactions}(\mathcal{T}, a_i)$
        $\mathcal{C}_{a_i} \leftarrow \text{Trans2Sentences}(\mathcal{T}_{a_i})$
        $\mathbf{E}^s_{a_i} \leftarrow \text{EncodeSemantic}(\mathcal{C}_{a_i}; \Theta_{\text{TLM}})$
        $\mathbf{E}^g_{a_i} \leftarrow \text{EncodeSimilarity}(\mathcal{C}_{a_i}, \mathcal{G}_w; \Theta_{\text{TASG}})$
        $\mathbf{E}_{a_i} \leftarrow \text{ConcatEmbedding}(\mathbf{E}^s_{a_i}, \mathbf{E}^g_{a_i})$
        $\mathbf{h}_{a_i} \leftarrow \text{FuseEmbed}(\mathbf{E}_{a_i}; \Theta_{\text{MAN}})$
      end
      ▷ Initialize AIG and farwad GNNs
      $\mathcal{G}_a^{\text{batch}} \leftarrow \text{ConstructAIG}(\{a_i\}, \mathcal{T})$
      $\mathbf{Z}_{\text{GCN}} \leftarrow \text{AIGForward}(\mathcal{G}_a^{\text{batch}}, \{\mathbf{h}_{a_i}\}; \Theta_{\text{AIG}})$
      $\mathbf{Z}_{\text{MAN}} \leftarrow \text{Predict}(\{\mathbf{h}_{a_i}\}; \Theta_{\text{MLP}})$
      ▷ Combine predictions and compute loss
      $\mathbf{Z}_{\text{Pred}} \leftarrow \lambda \mathbf{Z}_{\text{GCN}} + (1-\lambda)\mathbf{Z}_{\text{MAN}}$
      $\mathcal{L} \leftarrow \text{LossFunction}(\mathbf{Z}_{\text{Pred}}, \{y_i\})$
      ▷ Update model parameters
      $\Theta \leftarrow \Theta - \eta \nabla_\Theta \mathcal{L}$
    end
  end
return all parameters Θ

To reduce computational complexity and memory requirements, we introduce a batch update method to achieve synchronous mini-batch training for both the MAN and GCN. Specifically, we construct a dictionary to track the embeddings of all accounts in both models. In each iteration, we sample a mini-batch from phishing and normal accounts, compute their semantic embeddings, and update the corresponding node embeddings in the AIG through the dictionary.

To mitigate confounding biases, the model is fine-tuned to leverage shared representations for prediction. We jointly optimize both MAN predictions and GCN predictions to enhance overall performance. Specifically, we use the updated node embeddings to derive the GCN output, and after performing prediction interpolation, we calculate the cross-entropy loss for the current mini-batch. The loss function can be expressed as:

$$\mathcal{L} = -\frac{1}{B}\sum_{i=1}^{B}\left[y_i \log(\text{Pred}_i) + (1-y_i)\log(1-\text{Pred}_i)\right] \tag{19}$$

where B is the mini-batch size, y_i is the ground-truth label for sample i (1 for phishing account, 0 for normal account), and Pred_i is the predicted probability for sample i.

Table 2
Summary of three datasets.

Dataset	Nodes	Edges	Avg degree	Phisher
MulDiGraph	2,973,489	13,551,303	4.5574	1,165
B4E	597,258	11,678,901	19.5542	3,220
SPN	496,740	831,082	1.6730	5,619

During training, the transaction language model first initializes the embeddings of the account nodes in the AIG. The GCN then updates these node embeddings by aggregating information from neighboring nodes, capturing topological structure features. The joint optimization of the parameters of the MAN and GCN gradually enhances the complementary advantages of the language model and the graph model. The final model demonstrates excellent performance in detecting Ethereum phishing accounts.

4. Dataset review

As shown in Table 2, we utilized three datasets: MulDiGraph,[1] B4E,[2] and SPN. We process the dataset according to the methods in Section 3.1, 3.2 and 3.4 to obtain account transaction text, similarity information and transaction graph respectively.

4.1. MulDiGraph

This dataset is publicly available on the XBlock [43] platform and is a widely used dataset that was released in December 2020. It includes a large Ethereum transaction network obtained by performing a two-hop Breadth-First Search (BFS) from known phishing nodes. The dataset contains 2,973,489 nodes, 13,551,303 edges, and 1,165 phishing nodes, which contain basic transaction information such as transaction amount and timestamps.

4.2. B4E

This dataset was collected via an Ethereum node using Geth [31]. It covers transactions from January 1, 2017, to May 1, 2022, including 3,220 phishing accounts and 594,038 normal accounts. The dataset contains 328,261 transactions involving phishing accounts and 1,350,640 involving normal accounts. It is divided into four groups: phishing accounts, normal accounts, incoming transactions, and outgoing transactions.

4.3. Our dataset SPN

Second order Phishing Network, we created this dataset using the Etherscan API. Starting from the most recently identified all phishing nodes, we performed a two-hop BFS to gather information on their neighbors and extracted the first 100 transaction records for each involved node [44]. This dataset contains trading information prior to June 7, 2024, includes 5,619 phishing accounts and 491,121 normal accounts, with a total of 831,082 transaction edges. Compared to other datasets, SPN provides an up-to-date view of the Ethereum trading environment, focusing on recent phishing activities and network dynamics in a graph structure.

[1] xblock.pro/#/dataset/13
[2] https://github.com/git-disl/BERT4ETH

Table 3
The performances with our method and baseline methods on three datasets, and B-Acc is a Balanced Accuracy.

Method	MulDiGraph				B4E				SPN			
	Precision	Recall	F1	B-Acc	Precision	Recall	F1	B-Acc	Precision	Recall	F1	B-Acc
Role2Vec	0.4688	0.6976	0.5608	0.6511	0.5748	0.7958	0.6673	0.7507	0.4521	0.7059	0.5512	0.6391
Trans2Vec	0.7114	0.6944	0.7029	0.7768	0.2634	0.7043	0.3842	0.3598	0.3928	0.7381	0.5134	0.5838
GCN	0.2960	0.7513	0.4247	0.4289	0.5515	0.7508	0.6359	0.7228	0.5046	0.4973	0.5009	0.6266
GAT	0.2689	0.7917	0.4014	0.3577	0.4729	0.8348	0.6038	0.6848	0.5083	0.7720	0.6130	0.6993
SAGE	0.3571	0.3299	0.3430	0.5164	0.4589	0.5826	0.5134	0.6196	0.4557	0.5817	0.5110	0.6172
TSGN	0.6305	0.7148	0.6700	0.7526	0.6513	0.7954	0.7161	0.7912	0.6055	0.6793	0.6402	0.7290
GrabPhisher	0.7639	0.8369	0.7987	0.8537	0.7251	0.6138	0.6648	0.7487	0.6736	0.7929	0.7283	0.8003
BERT4ETH	0.4469	0.7344	0.5557	0.6400	0.7421	0.6125	0.6711	0.7530	0.7566	0.6713	0.7114	0.7817
ZipZap	0.4312	0.7128	0.5429	0.6247	0.7356	0.6112	0.6676	0.7506	0.7496	0.6733	0.7094	0.7804
Ours	**0.8919**	**0.9167**	**0.9041**	**0.9305**	**0.8158**	**0.8087**	**0.8123**	**0.8587**	**0.7962**	**0.8339**	**0.8146**	**0.8636**
Improv. (%)	16.8	9.5	13.2	9.0	9.9	−2.6	9.62	8.5	11.8	5.2	11.8	7.9

5. Experience

In this section, we present the experimental results of TLMG4Eth, an Ethereum phishing account detection model. We raised several research questions that we wanted to explore and answered them through experimental investigations, including comparison with common phishing detection methods, the validity and necessity of each module, proving the superior performance of our method in Ethereum phishing account detection.

- **RQ1**: How well does TLMG4Eth based on transaction semantics and transaction networks detect phishing scams compared to other baseline phishing scam detection methods?
- **RQ2**: Does the introduction of similarity embedding from the transaction attribute similarity graph by applying graph convolution effectively enhance the model?
- **RQ3**: How does the parameter λ that controls the weight of the language model and GNN model in joint training affect the performance of the model?
- **RQ4**: How does the combination of different GNN models and language models affect model performance?

TLMG4Eth is compared against several baselines including graph embedding methods (Role2Vec [45], Trans2Vec [25]), graph neural networks (GCN [46], GAT [47], SAGEConv [48]), and a Transformer-based method (BERT4ETH [31]). For practical considerations, we limit our analysis to the 100 most recent transactions per account. Our model configuration employs the BERT-base architecture as the language model, while both the transaction attribute similarity graph (TASG) and the account interaction graph (AIG) utilize two GCN layers. The joint training process is conducted with a batch sampling size of 64 and a learning rate of 1e−5. Unless otherwise specified, the trade-off parameter (λ) for joint training of TLM and AIG is set to 0.7. The TASG is constructed using TF-IDF scores, with an edge threshold of TF-IDF $\geq$ 0.2 to establish link relationships between nodes. The training process lasted for 20 epochs, taking approximately 3.5 h to complete. All training and inference experiments were conducted on an RTX 3090 with 24 GB of memory. For baseline methods, we adopt the original parameter settings for Trans2Vec and BERT4ETH as reported in their respective publications, while other baselines retain their default configurations without modifications.

5.1. Comparison with baselines (RQ1)

In this section, we present a comparative analysis of our proposed TLMG4Eth model with nine baseline methods. Table 3 and Fig. 4 summarizes the experimental results across the three datasets. While GrabPhisher achieves the highest F1 scores on MulDiGraph (0.7987) and SPN (0.7283), TSGN demonstrates the best baseline performance on B4E (0.7161). Nevertheless, TLMG4Eth consistently outperforms these top baselines on all three datasets, improving F1 scores by approximately 13.2%, 9.62%, and 11.8% on MulDiGraph, B4E, and SPN, respectively. TLMG4Eth also boosts Balanced Accuracy by 9.0%, 8.5%, and 7.9% over the strongest baselines in each dataset. The slight decrease in B4E recall (−2.61%) is offset by significant precision gains (7.37%), resulting in a net improvement in F1-score (9.62%). These substantial improvements highlight the efficacy of combining transaction language modeling with graph-based learning for phishing account detection.

We attribute these performance gains to two main factors: (i) By processing and extracting semantic features from transaction records, the language model component effectively captures subtle patterns of abnormal behaviors exhibited by phishing accounts, distinguishing them from legitimate ones. (ii) TLMG4Eth integrates the strengths of a language model and a graph neural network, enabling the model to leverage both semantic and structural information. This integration yields more expressive account representations, thereby enhancing classification accuracy.

It is worth noting that TLMG4Eth achieves a slightly larger improvement on MuDiGraph than on the other two datasets. We speculate that this is due to the data collection strategy for MuDiGraph (and partially for SPN), which employs breadth-first search expansions from phishing nodes, yielding more densely connected local transaction subgraphs. Such connectivity captures a richer set of topological relationships, enabling the model to more effectively leverage global structural cues.

Among the baseline methods, graph embedding-based algorithms (e.g., Role2Vec, Trans2Vec) exhibit relatively high recall but suffer from lower precision, suggesting a tendency to over-identify phishing nodes without sufficient discrimination for normal nodes. This overemphasis leads to elevated false positives. Additionally, these methods primarily focus on network structure, underutilizing intrinsic transaction attributes that can be especially significant in more structurally complex datasets like B4E.

Graph neural network (GNN)-based models also have constraints, particularly in how node features propagate through graph convolutions. As neighborhood aggregation deepens, nodes may converge toward the majority class (non-phishing), making phishing nodes more difficult to distinguish [49,50]. By contrast, our TLMG4Eth framework supplements graph convolution with a transaction language model, continuously refining node representations with semantic cues. This design lessens the risk that phishing nodes become indistinguishable as graph convolution layers increase.

Although the Transformer-based BERT4ETH and ZipZap baseline is competitive, especially in sequential transaction analysis, its limited capacity to encode structural topological information constrains its ability to capture higher-level relational patterns among accounts. Furthermore, BERT4ETH's numerical treatment of transaction features does not incorporate the nuanced semantic perspective on account behavior, potentially overlooking critical indicators of phishing activity. In contrast, TLMG4Eth's integrated semantic and structural representation allows it to detect subtle anomalies more robustly.

While TSGN and GrabPhisher demonstrate strong baseline performance, they exhibit key limitations in phishing detection. TSGN, which

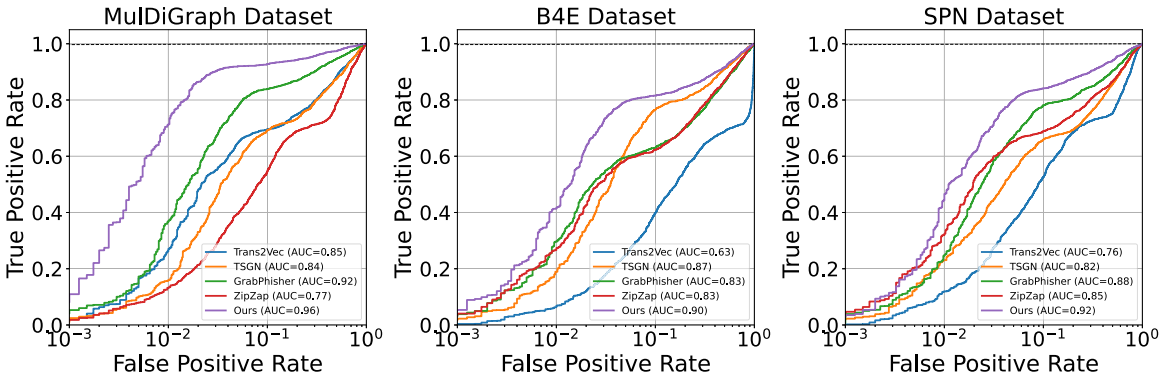

Fig. 4. ROC curves of the baseline model and TLMG4Eth on different datasets.

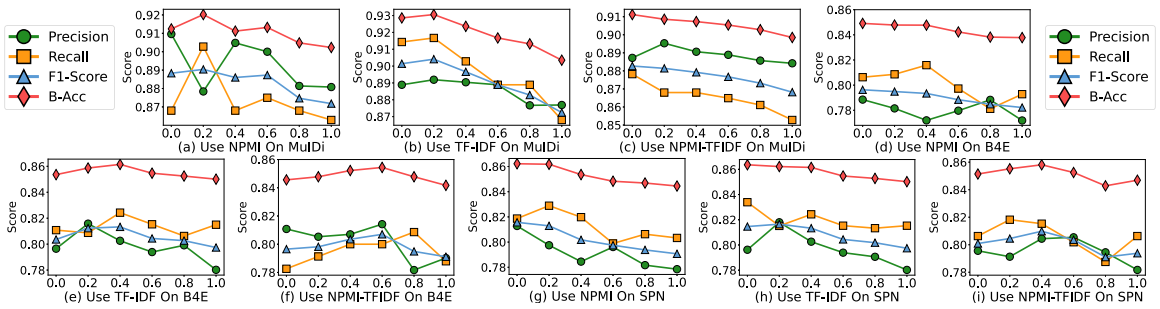

Fig. 5. Performance of various TASG construction methods under varying threshold θ.

Table 4
Performance comparison of TLM combined with different versions of TASG versus TLM alone.

Enhancer	MulDiGraph				B4E				SPN			
	Precision	Recall	F1	B-Acc	Precision	Recall	F1	B-Acc	Precision	Recall	F1	B-Acc
w/o	0.8776	0.8731	0.8753	0.9061	0.7807	0.7850	0.7825	0.8374	0.7902	0.7879	0.7911	0.8417
NPMI-TFIDF	**0.8953**	0.8680	0.8814	0.9086	0.8053	0.7913	0.7982	0.8478	0.7913	0.8182	0.8045	0.8551
Improv. (%)	**1.77**	−0.51	0.61	0.25	2.46	0.63	1.57	1.04	0.11	3.03	1.34	1.34
NPMI	0.8784	0.9028	0.8904	0.9202	0.7815	0.8086	0.7949	0.8478	**0.7975**	0.8289	0.8129	0.8618
Improv. (%)	0.08	2.97	1.51	1.40	0.08	2.36	1.24	1.04	**0.73**	4.10	2.18	2.01
TF-IDF	0.8919	**0.9167**	**0.9041**	**0.9306**	**0.8158**	**0.8087**	**0.8123**	**0.8587**	0.7962	**0.8339**	**0.8146**	**0.8636**
Improv. (%)	1.43	**4.36**	**2.88**	**2.45**	**3.51**	**2.37**	**2.98**	**2.13**	0.60	**4.60**	**2.35**	**2.19**

relies on static graph motifs, performs well on B4E (F1 = 0.7161) but struggles against adaptive phishing tactics that mimic legitimate transaction structures. Conversely, GrabPhisher excels on MulDiGraph (F1 = 0.7987) but falters on sparser datasets like B4E (F1 = 0.6648), where its dependence on dense phishing clusters limits its effectiveness. In contrast, TLMG4Eth consistently outperforms both models by integrating semantic and structural signals, ensuring more robust phishing detection across diverse datasets.

5.2. Ablation study of attribute similarity graph (RQ2)

When modeling the transaction semantics of an account, the basic BERT model can only consider the transaction sequence information of the input single account, and does not consider the semantic information of other accounts. However, this global semantic information is crucial to gain insight into abnormal transactions and enhance the generalization ability of the model. Therefore, we introduce the transaction attribute similarity graph(TASG) into BERT to enhance the global awareness of the language model. In this section, we further explore the impact of the key component transaction attribute similarity graph in TLMG4Eth on the prediction performance of the model and conduct ablation experiments on it. We conduct experiments on three datasets respectively. The experimental results are presented in Fig. 5 and Table 4. NPMI represents that we use the normalized pointwise mutual information to construct the TASG, and concatenate the features obtained by graph convolution with the semantic features extracted by BERT for deep Multi-Head Attention Network learning. TF-IDF means we use term frequency-inverse document frequency to build TASG; NPMI-TFIDF means that we used both NPMI and TF-IDF composition; The notation "w/o" means that we do not introduce transaction attribute similarity information into the language model and only use pure BERT to extract features. These experiments are all done under the trade-off parameter $\lambda = 0.7$ and jointly training the model as GCN using edge features.

Overall, regardless of the method employed, the introduction of transaction attribute similarity graphs improved model performance. Notably, when using different methods to construct similar graphs, a lower threshold theta value between 0 to 0.4 will result in better performance of the model. This is because higher theta filters out most of the mutual information of words, reducing the structure of the vocabulary graph, making it difficult to obtain high-quality topological and similarity features through convolution of the vocabulary graph.

The model that incorporating the similarity information of transaction attributes extracted by TF-IDF features achieves the most significant improvement. Compared with BERT without introducing any similarity information, it improves the F1 Scores by about 2.88%,

Table 5
Effect of varying trade-off parameter λ between TLM and AIG on model.

λ	MulDiGraph				B4E				SPN			
	Precision	Recall	F1	B-Acc	Precision	Recall	F1	B-Acc	Precision	Recall	F1	B-Acc
0	0.8918	0.8782	0.8849	0.9125	0.7648	0.8003	0.7820	0.8386	0.7856	0.8103	0.7972	0.8499
0.1	0.8788	0.8832	0.8810	0.9111	0.7741	0.8072	0.7903	0.8447	0.7949	0.8142	0.8040	0.8546
0.2	0.8964	0.8782	0.8872	0.9137	**0.7963**	0.7822	0.7892	0.8411	**0.8174**	0.7992	0.8082	0.8550
0.3	**0.9014**	0.8889	0.8951	0.9201	0.7716	0.8163	0.7933	0.8477	0.7923	0.8333	0.8123	0.8620
0.4	0.9000	0.8750	0.8873	0.9132	0.7702	0.8134	0.7912	0.8460	0.7910	0.8304	0.8102	0.8603
0.5	0.8974	0.8883	0.8929	0.9188	0.7875	0.7885	0.7878	0.8411	0.8127	0.8055	0.8091	0.8563
0.6	0.9028	0.9028	0.9028	0.9271	0.7445	**0.8259**	0.7830	0.8421	0.7693	0.8429	0.8044	0.8582
0.7	0.8919	**0.9167**	**0.9041**	**0.9306**	0.7824	0.7947	0.7885	0.8421	0.8033	0.8117	0.8075	0.8562
0.8	0.8904	0.9028	0.8966	0.9236	0.7817	0.8086	**0.7949**	**0.8478**	0.7821	**0.8491**	**0.8142**	**0.8654**
0.9	0.8767	0.8889	0.8828	0.9132	0.7676	0.8026	0.7847	0.8406	0.7712	0.8397	0.8040	0.8576
1	0.3172	0.8194	0.4574	0.4687	0.5742	0.7477	0.6495	0.7352	0.5742	0.7477	0.6495	0.7352

2.98% and 2.35% in the MulDiGraph, B4E and SPN, respectively. The B-Acc index is improved by about 2.45%, 2.13% and 2.19%, respectively. The other two methods NPMI and NPMI-TFIDF also have varying degrees of improvement compared with only using BERT, which shows that our method of integrating transaction attribute similarity features into the semantic features extracted by the language model effectively enhances the detection ability of the model.

One possible reason why TF-IDF improves the model more than NPMI and NPMI-TFIDF is that: TF-IDF is commonly used in NLP to measure the importance of terms in a document [51]. If a term appears many times in a document, but not many times in other documents, then this term is important to this document and conveys the key information of this document [52]. In the Ethereum transaction scenario, phishing accounts and their transaction records appear frequently in phishing scams, but account for a small proportion of the total transaction information, so the TF-IDF value of the words involved in phishing in the transaction language will be higher, that is, these words can express key information of phishing accounts, which provides key global features for the language model. Thus, the global perception ability of the model is greatly improved.

NPMI-based methods focus on co-occurrence probabilities. NPMI can be understood as a pre-clustering of corpus information, where strongly correlated words are grouped together. If two words have a higher probability of co-occurrence in the trading language, they are more closely related and have a higher degree of association. In the scenario of normal account transactions in Ethereum, most of the transaction records structure are similar, and NPMI will pay more attention to the transaction semantics of normal accounts. Therefore, compared with TF-IDF information, the global similarity features of phising accounts provided by NPMI are limited.

5.3. Impact of the trade-off parameter (RQ3)

In this section, we discuss in detail the effect of the trade-off parameter λ of the language model TLM and GNNs model on the performance of the model in joint training. These experiments were done under TLM using NPMI as the similarity feature and GCN-e as the joint training model. When λ is close to 0, the model will be biased toward the prediction results of the language model to optimize the model. However, when λ is close to 1, the model will be biased toward the decision update model parameters of the GNNs model based on semantic embedding. We have proved through extensive experiments that the optimal trade-off parameters of the model on different datasets are different, which may be caused by the different distribution of the datasets. However, in general, the best performance of TLMG4Eth on each dataset occurs at high values of λ, that is, when the contribution of GNNs model is relatively large, TLMG4Eth has better classification performance.

The specific experimental results are shown in Table 5. On the MuDiGraph, B4E and SPN datasets, the F1 Scores and B-Acc of the model is the highest when λ is equal to 0.7, 0.8 and 0.8, respectively. The highest F1-Score was 90.41%, 79.49% and 81.42%. The highest B-Acc was 93.06%, 84.78% and 86.54%. Across all three datasets, the results of joint training outperform the cases where $\lambda = 0$ or 1. This indicates that joint training is more effective than using either approach independently. Taking the results on the MuDiGraph dataset as an example, when $\lambda = 0.7$, the F1 Scores and B-Acc of the model are improved by about 1.92% and 1.81% respectively compared with $\lambda = 0$, which shows that the introduction of the GNNs model for joint training effectively improves the classification performance of the model.

Although the model performs better when the trade-off parameter λ is relatively high, this does not mean that increasing the weight of the GNNs model infinitely or even relying on the GNNs model to make the decision will lead to better performance. We compare the training loss curves of the model under different λ. Fig. 6 shows that when λ is set to 0.9, compared with other λ values, the training loss of the model converges to a higher value. When λ is set to 1, the loss hardly decreases as training progresses, and the performance of the model degrades substantially, even on par with the performance of GCN in the baseline method. One possible reason is that when λ is too high, the decision results dominated by GNNs have limited reference significance to TLM, and TLM cannot optimize the extraction of transaction semantic features in joint training. Moreover, TLM still updates the embeddings of intermediate transaction graph nodes with each round of training, which makes GNNs continue to receive lower-quality semantic embeddings, ultimately making the overall model difficult to optimize. Our research proves that an appropriately high λ value can well integrate the complementary advantages of language model and graph model in joint training, thus obtaining excellent performance.

5.4. Different GNN model combination (RQ4)

In this section, we explore the performance differences of various graph representation learning algorithms applied to AIG. We evaluate four graph learning algorithms, each jointly trained with TLM, including: Graph Convolutional Network (GCN) without edge features, GCN with edge features (GCNe), Graph Attention Network (GAT), and GraphSAGE convolution (SAGEconv). For fair comparison, we set the number of conv layers to 2 for all GNNs, the number of heads for GAT to 8, and the aggregation operation of mean for SAGEconv. In addition, we adopt TF-IDF similarity graph enhancement of transaction attributes for TLM, and keep the rest Settings as default, and train for 50 epochs on three different datasets respectively.

The experimental results, presented in Table 6, demonstrate that GCNe, which incorporates edge features, achieves the best performance on the three datasets. Compared to the standalone multihead attention network(MAN), the MAN + GCNe model shows significant improvements across all three datasets. Specifically, it enhances the F1 Scores by approximately 2.37%, 2.00%, and 1.28%, while the B-Acc metric improves by about 2.06%, 1.56%, and 1.17% respectively.

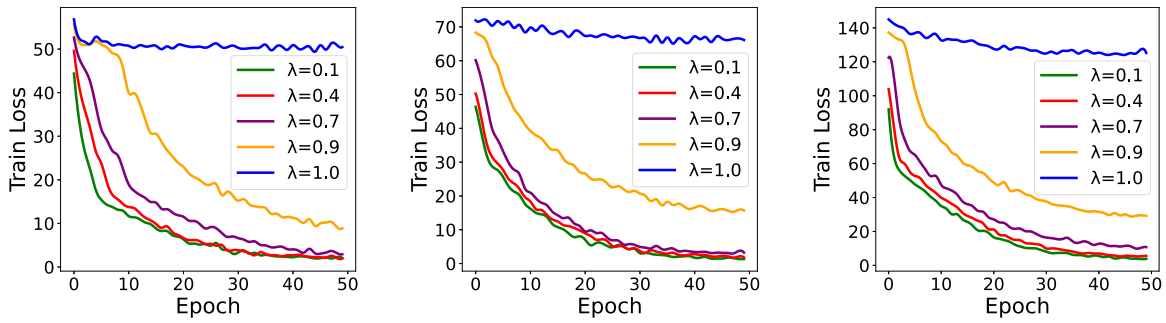

Fig. 6. Training Loss vs Epoch on MulDiGraph, B4E and SPN datasets with different trade-off parameter λ.

Table 6
Performance of TLM combined with various GNN.

Combination	MulDiGraph				B4E				SPN			
	Precision	Recall	F1	B-Acc	Precision	Recall	F1	B-Acc	Precision	Recall	F1	B-Acc
TLM Only	0.8827	0.8782	0.8804	0.9099	0.8002	0.7841	0.7923	0.8431	0.7980	0.8056	0.8018	0.8518
TLM+GCN	0.8667	0.9028	0.8844	0.9167	0.8074	0.7974	0.8027	0.8511	0.7905	0.8128	0.8015	0.8526
Improv. (%)	−1.60	2.46	0.40	0.68	0.72	1.33	1.04	0.80	−0.75	0.72	−0.03	0.07
TLM+GAT	0.8767	0.8889	0.8828	0.9132	0.8086	0.7818	0.7949	0.8446	**0.8107**	0.7984	0.8045	0.8526
Improv. (%)	−0.60	1.07	0.24	0.33	0.84	−0.23	0.26	0.15	**1.27**	−0.72	0.27	0.08
TLM+SAGE	0.8866	0.8731	0.8798	0.9086	0.8032	0.8037	0.8032	0.8526	0.7813	0.8182	0.7993	0.8518
Improv. (%)	0.39	−0.51	−0.06	−0.13	0.30	1.96	1.09	0.95	−1.68	1.26	−0.25	0.00
TLM+GCNe	**0.8919**	**0.9167**	**0.9041**	**0.9306**	**0.8158**	**0.8087**	**0.8123**	**0.8587**	0.7962	**0.8339**	**0.8146**	**0.8636**
Improv. (%)	**0.92**	**3.85**	**2.37**	**2.06**	**1.56**	**2.46**	**2.00**	**1.56**	−0.19	**2.83**	**1.28**	**1.17**

This together with the experiment in RQ3 proves that our proposed TLM and AIG joint training method effectively improves the phishing account detection performance of the model. Notably, the SAGEconv and GCN models exhibit a considerable performance gap compared to GCNe in joint training, even diminishing the effectiveness of MAN. This may be due to the failure of the model to model the relationship between nodes and their neighbors, and the graph convolution operation does not assign different weights according to the importance of nodes, so that nodes will aggregate useless or even interfere with the features of account classification, and ultimately affect the classification performance.

Although GAT captures the relationship and feature relationship between nodes and their neighbors through the self-attention mechanism, it only relies on the connection relationship between nodes to calculate the self-attention coefficient to assign weights to neighbors, rather than explicitly using edge features [53,54]. In the context of the Ethereum transaction network being a sparse network, this feature that GAT relies on node connectivity makes it more difficult to accurately capture the connections between nodes. In contrast, GCNe using edge features explicitly models the weights between nodes and their neighbors based on Ethereum transaction records, uses edge features to aggregate messages from limited neighbors in sparse graphs, and successfully combines transaction semantic information and network topology information, thereby achieving excellent performance.

6. Limitation

While TLMG4Eth demonstrates significant improvements in Ethereum fraud detection, several limitations persist that should be acknowledged and addressed in future research:

Data Temporal Coverage: The SPN dataset used focuses on phishing patterns up to June 2024, limiting the model's adaptability to emerging fraud tactics. A more longitudinal dataset could improve generalizability and robustness.

Computational Complexity: The integration of 12-layer Transformers with GCNs introduces significant computational overhead, particularly for memory-intensive attention mechanisms. This poses scalability challenges for real-time detection, especially given the large scale of Ethereum's transaction network. Optimization strategies are needed for large-scale deployment.

Semantic Drift Risk: The static pre-training of the TLM model may lead to outdated semantics as phishing tactics evolve. Without dynamic learning to adapt to new tactics, the model may struggle to detect novel fraud types over time.

Multi-chain Generalization: Our approach is tested only on Ethereum, limiting its applicability to other blockchains, such as Bitcoin. Future work could explore methods to generalize the model across different blockchain structures.

7. Conclusion

In this paper, we introduced TLMG4Eth, a novel approach that integrates transaction language models with graph-based methods to capture semantic, similarity, and structural features of transaction data in Ethereum. Our work represents the first attempt to utilize language models to address the issue of unclear transaction semantics, and we pioneered the modeling of transaction similarity. After using an attention network to fuse semantic and similarity information, we proposed the construction of an account interaction graph to capture the structural information of the account transaction network. Furthermore, we developed a method for jointly training the attention network and the account structure graph to integrate information from all stages. Our approach has demonstrated significant improvements, achieving performance gains of approximately 10% or more across three datasets compared to the current state-of-the-art methods. These empirical results provide strong evidence for the effectiveness of our proposed methodology, highlighting the potential of combining linguistic, semantic, and structural analysis in blockchain analytics and fraud detection. As part of interesting future work, we plan to explore a wider range of transaction semantic language models, improve pre-training objectives, and study more effective fusion networks for fraud detection tasks on other blockchains.

CRediT authorship contribution statement

Jianguo Sun: Project administration, Methodology, Investigation. **Yifan Jia:** Writing – original draft, Software, Data curation. **Yanbin Wang:** Supervision, Methodology. **Ye Tian:** Formal analysis, Conceptualization. **Sheng Zhang:** Visualization, Methodology.

Declaration of competing interest

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

Acknowledgments

Our research is supported by the Natural Science Foundation of China, Grant No. 62302358.

Data availability

The code and data for our research are available at: https://github.com/lincozz/TLmGNN.

References

[1] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, H. Wang, Blockchain challenges and opportunities: A survey, Int. J. Web Grid Serv. 14 (4) (2018) 352–375.
[2] Z. Zheng, J. Su, J. Chen, D. Lo, Z. Zhong, M. Ye, Dappscan: building large-scale datasets for smart contract weaknesses in dapp projects, IEEE Trans. Softw. Eng. (2024).
[3] G. Wood, et al., Ethereum: A secure decentralised generalised transaction ledger, 2014, pp. 1–32, Ethereum project yellow paper 151.
[4] S. Wu, D. Wang, J. He, Y. Zhou, L. Wu, X. Yuan, Q. He, K. Ren, Defiranger: Detecting price manipulation attacks on defi applications, 2021, arXiv preprint arXiv:2104.15068.
[5] D. Wang, S. Wu, Z. Lin, L. Wu, X. Yuan, Y. Zhou, H. Wang, K. Ren, Towards a first step to understand flash loan and its applications in defi ecosystem, in: Proceedings of the Ninth International Workshop on Security in Blockchain and Cloud Computing, 2021, pp. 23–28.
[6] S. Werner, D. Perez, L. Gudgeon, A. Klages-Mundt, D. Harz, W. Knottenbelt, Sok: Decentralized finance (defi), in: Proceedings of the 4th ACM Conference on Advances in Financial Technologies, 2022, pp. 30–46.
[7] L. Zhou, K. Qin, A. Cully, B. Livshits, A. Gervais, On the just-in-time discovery of profit-generating transactions in defi protocols, in: 2021 IEEE Symposium on Security and Privacy, SP, IEEE, 2021, pp. 919–936.
[8] S. Li, G. Gou, C. Liu, G. Xiong, Z. Li, J. Xiao, X. Xing, TGC: Transaction graph contrast network for ethereum phishing scam detection, in: Proceedings of the 39th Annual Computer Security Applications Conference, 2023, pp. 352–365.
[9] R. Kaur, D. Gabrijelčič, T. Klobučar, Artificial intelligence for cybersecurity: Literature review and future research directions, Inf. Fusion 97 (2023) 101804.
[10] Chainalysis, 2023 crypto crime trends: Illicit cryptocurrency volumes reach all-time highs amid surge in sanctions designations and hacking, 2023, https://www.chainalysis.com/blog/2023-crypto-crime-report-introduction/. (Accessed 1 December 2023).
[11] A. Vaswani, Attention is all you need, 2017, arXiv preprint arXiv:1706.03762.
[12] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, AI Open 1 (2020) 57–81.
[13] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S.Y. Philip, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. 32 (1) (2020) 4–24.
[14] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks? 2018, arXiv preprint arXiv:1810.00826.
[15] R. Nogueira, Z. Jiang, J. Lin, Document ranking with a pretrained sequence-to-sequence model, 2020, arXiv preprint arXiv:2003.06713.
[16] Z. Tüske, G. Saon, K. Audhkhasi, B. Kingsbury, Single headed attention based sequence-to-sequence model for state-of-the-art results on switchboard, 2020, arXiv preprint arXiv:2001.07263.
[17] A. Cohan, I. Beltagy, D. King, B. Dalvi, D.S. Weld, Pretrained language models for sequential sentence classification, 2019, arXiv preprint arXiv:1909.04054.
[18] C. Zhang, The analysis of the risks and improvements of ERC20 tokens, Highlights Sci. Eng. Technol. 39 (2023) 1093–1097.
[19] L. Zhou, K. Qin, C.F. Torres, D.V. Le, A. Gervais, High-frequency trading on decentralized on-chain exchanges, in: 2021 IEEE Symposium on Security and Privacy, SP, IEEE, 2021, pp. 428–445.
[20] Z. Li, J. Li, Z. He, X. Luo, T. Wang, X. Ni, W. Yang, X. Chen, T. Chen, Demystifying defi mev activities in flashbots bundle, in: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, 2023, pp. 165–179.
[21] K. Qin, L. Zhou, A. Gervais, Quantifying blockchain extractable value: How dark is the forest? in: 2022 IEEE Symposium on Security and Privacy, SP, IEEE, 2022, pp. 198–214.
[22] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 855–864.
[23] R. Tan, Q. Tan, P. Zhang, Z. Li, Graph neural network for ethereum fraud detection, in: 2021 IEEE International Conference on Big Knowledge, ICBK, 2021, pp. 78–85, http://dx.doi.org/10.1109/ICKG52313.2021.00020.
[24] Q. Yuan, B. Huang, J. Zhang, J. Wu, H. Zhang, X. Zhang, Detecting phishing scams on ethereum based on transaction records, in: 2020 IEEE International Symposium on Circuits and Systems, ISCAS, 2020, pp. 1–5, http://dx.doi.org/10.1109/ISCAS45731.2020.9180815.
[25] J. Wu, Q. Yuan, D. Lin, W. You, W. Chen, C. Chen, Z. Zheng, Who are the phishers? phishing scam detection on ethereum via network embedding, IEEE Trans. Syst. Man, Cybern.: Syst. 52 (2) (2020) 1156–1166.
[26] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 701–710.
[27] L. Wang, M. Xu, H. Cheng, Phishing scams detection via temporal graph attention network in ethereum, Inf. Process. Manage. 60 (4) (2023) 103412.
[28] X. Zhou, W. Yang, X. Tian, Detecting phishing accounts on ethereum based on transaction records and EGAT, Electronics 12 (4) (2023) 993.
[29] S. Li, R. Wang, H. Wu, S. Zhong, F. Xu, SIEGE: Self-supervised incremental deep graph learning for ethereum phishing scam detection, in: Proceedings of the 31st ACM International Conference on Multimedia, 2023, pp. 8881–8890.
[30] C. Wu, J. Chen, Z. Zhao, K. He, G. Xu, Y. Wu, H. Wang, H. Li, Y. Liu, Y. Xiang, Tokenscout: Early detection of ethereum scam tokens via temporal graph learning, in: Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, 2024, pp. 956–970.
[31] S. Hu, Z. Zhang, B. Luo, S. Lu, B. He, L. Liu, Bert4eth: A pre-trained transformer for ethereum fraud detection, in: Proceedings of the ACM Web Conference 2023, 2023, pp. 2189–2197.
[32] S. Hu, T. Huang, K.-H. Chow, W. Wei, Y. Wu, L. Liu, ZipZap: Efficient training of language models for large-scale fraud detection on blockchain, in: Proceedings of the ACM on Web Conference 2024, WWW '24, Association for Computing Machinery, New York, NY, USA, 2024, pp. 2807–2816, http://dx.doi.org/10.1145/3589334.3645352.
[33] J. Wang, P. Chen, X. Xu, J. Wu, M. Shen, Q. Xuan, X. Yang, Tsgn: Transaction subgraph networks assisting phishing detection in ethereum, 2022, arXiv preprint arXiv:2208.12938.
[34] S. Li, G. Gou, C. Liu, C. Hou, Z. Li, G. Xiong, TTAGN: Temporal transaction aggregation graph network for ethereum phishing scams detection, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 661–669.
[35] A.Q. Md, S.S.S. Narayanan, H. Sabireen, A.K. Sivaraman, K.F. Tee, A novel approach to detect fraud in Ethereum transactions using stacking, Expert Syst. 40 (7) (2023) e13255.
[36] S. Dutta, A. Sharma, J. Rajgor, Ethereum fraud prevention: A supervised learning approach for fraudulent account recognition, in: 2024 1st International Conference on Trends in Engineering Systems and Technologies, ICTEST, IEEE, 2024, pp. 1–8.
[37] J. Zhang, H. Sui, X. Sun, C. Ge, L. Zhou, W. Susilo, GrabPhisher: Phishing scams detection in ethereum via temporally evolving GNNs, IEEE Trans. Serv. Comput. (2024).
[38] J. Devlin, Bert: Pre-training of deep bidirectional transformers for language understanding, 2018, arXiv preprint arXiv:1810.04805.
[39] C.E. Shannon, A mathematical theory of communication, Bell Syst. Tech. J. 27 (3) (1948) 379–423.
[40] J. Ramos, et al., Using tf-idf to determine word relevance in document queries, in: Proceedings of the First Instructional Conference on Machine Learning, 242, (1) 2003, pp. 29–48, Citeseer.
[41] Z. Lu, P. Du, J.-Y. Nie, VGCN-BERT: augmenting BERT with graph embedding for text classification, in: Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part I 42, Springer, 2020, pp. 369–382.
[42] S. Zhang, H. Tong, J. Xu, R. Maciejewski, Graph convolutional networks: a comprehensive review, Comput. Soc. Netw. 6 (1) (2019) 1–23.
[43] L. Chen, J. Peng, Y. Liu, J. Li, F. Xie, Z. Zheng, XBLOCK Blockchain Datasets: InPlusLab ethereum phishing detection datasets, 2019, http://xblock.pro/ethereum/.
[44] P. Zheng, Z. Zheng, J. Wu, H.-N. Dai, Xblock-eth: Extracting and exploring blockchain data from ethereum, IEEE Open J. Comput. Soc. 1 (2020) 95–106.
[45] N.K. Ahmed, R.A. Rossi, J.B. Lee, T.L. Willke, R. Zhou, X. Kong, H. Eldardiry, role2vec: Role-based network embeddings, Proc. DLG KDD (2019) 1–7.

[46] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2016, arXiv preprint arXiv:1609.02907.
[47] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, 2017, arXiv preprint arXiv:1710.10903.
[48] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Adv. Neural Inf. Process. Syst. 30 (2017).
[49] X. Li, Z. Fan, F. Huang, X. Hu, Y. Deng, L. Wang, X. Zhao, Graph neural network with curriculum learning for imbalanced node classification, Neurocomputing 574 (2024) 127229.
[50] G. Du, J. Zhang, M. Jiang, J. Long, Y. Lin, S. Li, K.C. Tan, Graph-based class-imbalance learning with label enhancement, IEEE Trans. Neural Netw. Learn. Syst. 34 (9) (2021) 6081–6095.
[51] W. Cunha, V. Mangaravite, C. Gomes, S. Canuto, E. Resende, C. Nascimento, F. Viegas, C. França, W.S. Martins, J.M. Almeida, et al., On the cost-effectiveness of neural and non-neural approaches and representations for text classification: A comprehensive comparative study, Inf. Process. Manage. 58 (3) (2021) 102481.
[52] A. Helan, Z.N. Sultani, Topic modeling methods for text data analysis: a review, in: AIP Conference Proceedings, 2457, (1) AIP Publishing, 2023.
[53] F. Errica, M. Podda, D. Bacciu, A. Micheli, A fair comparison of graph neural networks for graph classification, 2019, arXiv preprint arXiv:1912.09893.
[54] Y. Wang, J. Zhang, S. Guo, H. Yin, C. Li, H. Chen, Decoupling representation learning and classification for gnn-based anomaly detection, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 1239–1248.