TEGDetector: A Phishing Detector That Knows Evolving Transaction Behaviors

Haibin Zheng, Minying Ma, Haonan Ma, Jinyin Chen, Member, IEEE, Haiyang Xiong, and Zhijun Yang

Abstract—Recently, phishing scams have posed a significant threat to blockchains. Phishing detectors direct their efforts in hunting phishing addresses. Most of the detectors extract target addresses’ transaction behavior features by random walking or constructing static subgraphs. The random walking methods, unfortunately, usually miss structural information due to limited sampling sequence length, while the static subgraph methods tend to ignore temporal features lying in the evolving transaction behaviors. More importantly, their performance undergoes severe degradation when the malicious users intentionally hide phishing behaviors. To address these challenges, we propose TEGDetector, a dynamic graph classifier that learns the evolving behavior features from transaction evolution graphs (TEGs). First, we cast the transaction series into multiple time slices, capturing the target address’s transaction behaviors in different periods. Then, we provide a fast nonparametric phishing detector (FD) to narrow down the search space of suspicious addresses. Finally, TEGDetector utilizes adaptively learned time coefficient to pay distinct attention to different periods, which provides several novel insights. Extensive experiments on the large-scale Ethereum transaction dataset demonstrate that the proposed method achieves state-of-the-art (SOTA) detection performance. The code of TEGDetector is open sourced at https://github.com/Seaocn/TEGDetector.

Index Terms—Dynamic graph classification, phishing detector, robustness.

NOMENCLATURE

A. Abbreviations
TEGs Transaction evolution graphs.
FD Fast nonparametric phishing detector.
EP-Extractor Evolution feature extraction.
T-EDGE Temporal weighted multidigraph embedding.
MCGC Multichannel graph classification model.

B. Notation
\[ G = (V, E) \] Input graph \( G \) with nodes \( V \) and edges \( E \).
\( A \) Adjacency matrix.
\( X \) Node feature matrix.
\( G, G' \) Dynamic graph, a deception dynamic graph.
\( G_{train}, G_{test} \) Training/dynamic graph classification dataset.
\( y_i \) \( i \)th graph corresponding true label.
\( f_{\theta}(\cdot) \) Graph classification model with parameters \( \theta \).
\( \sigma(\cdot) \) Activation function.
\( z_t, r_t \) \( r \)th slice update gate, the \( r \)th slice reset gate.
\( C_t \) \( t \)th slice cluster assignment matrix.
\( A^{i+1}_{pool} \) (\( i + 1 \))th pooling adjacency matrix.
\( Z^{i+1}_{pool} \) (\( i + 1 \))th pooling node-level matrix.
\( Y \) Category set of the graphs.

I. INTRODUCTION

A decentralized and distributed public ledger, blockchain technology [1] has enjoyed great success in various fields, for example, finance, technology, and culture [1]. Cryptocurrency [2], [3], undoubtedly, is one of the most profound applications of blockchain. As the largest blockchain platform supporting smart contracts, Ethereum now holds cryptocurrencies worth more than $39.3 billion dollars. Unfortunately, the decentralization of blockchain also breeds numerous financial scams [4], [5], [6], [7], [8]. Chainalysis1 has reported that phishing scams, which accounted for 38.7% of all Ethereum scams [9], stole $34 million from Ethereum platform in 2018. Phishing refers to impersonating a website of an honest firm, which obtains the users’ sensitive information and money via phishing websites. Recently, phishing scams are reported every year, and they become even more sophisticated.

1A provider of investigation and risk management software for virtual currencies.
As a result, detecting phishing addresses on blockchain has attracted widespread attention. Fundamentally, phishing detection aims to learn a mapping function that bridges their historical transaction behaviors to a binary output y, where y = 0 denotes that the target address is normal address, and y = 1 represents it is phishing address. A family of works \[10\], \[11\] utilized manually extracted features to capture the target user’s transaction behaviors. Unfortunately, these models require an incisive data understanding, leading to unsatisfactory results.

Recent approaches explored different sorts of graph embedding algorithms to address the issue. Walking-based detectors \[12\], \[13\], \[14\], \[15\], \[16\], \[17\] direct their efforts at adopting random walking to characterize the temporal evolution between transactions. Subgraph-based detectors \[16\], \[18\], \[19\], \[20\] usually described the target address’s transaction pattern through a static subgraph. Specifically, they first construct the transactions of the target address and its neighbors in all periods into a static subgraph, then built upon the success of graph neural networks (GNNs) to learn the spatial graph structure from the static subgraphs.

All in all, we summarized the following challenges.

1) Unbalanced data distribution: Due to the unbalanced distribution ratio of normal nodes and phishing nodes, the model is prone to underfitting if the unprocessed data is directly used for training.

2) Space-time imbalance: The trading patterns of phishing nodes exhibit a dynamic nature, making it challenging to comprehensively capture spatial and temporal features. Conventional works tend to focus on either structural or temporal information for phishing address detection. They fail to integrate them toward a more precise and robust phishing address detector. This motivates us to consider whether we can combine and balance them to approach better detection performance. Due to the incompleteness of the structural information caused by the limited sampling sequence length, it is difficult for the walking strategy to achieve such a balance. We speculate that one viable approach is to construct multiple transaction subgraphs for a target address, where each subgraph characterizes the transaction topology within a temporal period. We term the transaction subgraphs as dynamic subgraphs. Taking the transaction graphs of two Ethereum users as an example, we apply the static subgraph construction method in MCGC \[19\] and extend it to the dynamic subgraph construction. As shown in Fig. 1(a), users \(u_1\) and \(u_2\) are with similar static subgraphs but actually they have significantly different transaction patterns, indicating that the dynamic subgraphs can frame both structural and temporal behavior cues.

3) Masquerading of addresses: Phishing nodes often remain concealed, leading to a lack of robustness in certain existing detection methods. Researches \[21\], \[22\], \[23\] on the vulnerability of the graph analysis methods reveal potential security issues in blockchain phishing detectors. Intuitively, the phishers may bypass the detection by transacting to specific addresses. To verify the robustness of existing phishing detectors, we randomly add transactions between the first- and second-order neighboring addresses of 200 verified Ethereum phishing addresses. Fig. 1(b) shows that the detection accuracies of state-of-the-art (SOTA) approaches such as Trans2vec \[14\], T-EDGE \[15\], identity inference on blockchain using GNN (\(I^2BGNN\)) \[18\], and MCGC \[19\] reduce significantly as the proportion of new transactions increases.

To overcome the above challenges, we propose TEGDetector, a dynamic graph classifier that learns the evolving behavior features from TEGs. Specifically, to tackle challenge (1), inspired by some methods \[24\], \[25\], \[26\], during data processing, we under-sample the normal addresses to adjust the sample distribution, which is beneficial to the training of the model. To address challenge (2), we construct a series of TEGs for multiple time slices, which have the key advantage of retaining both the spatial structural and temporal information. At last, to tackle challenge (3), we propose TEGDetector that serves to capture target addresses behavior features. The TEGDetector is composed of graph convolutional layers and gate recurrent unit (GRU), which capture the topology structure and dynamic evolution characteristics of the network, respectively. Specifically, we introduce adaptive time coefficients to comprehensively balance the user’s behavior features in all periods, rather than using only the one in the most recent period. This benefits exploring the crucial factors of phishing detection and helps TEGDetector identify possible malicious deception. It is worth noting that the time coefficient we are referring to here is different from the attention coefficient in this article \[27\]. Our approach is oriented toward time, emphasizing the importance of different temporal dimensions.
Graph attention networks (GATs) [27] are oriented toward the graph’s structure, specifically in allocating varying importance to different nodes within different neighborhoods when dealing with varying neighborhood sizes. Our contributions may be summarized as follows.

1) To the best of our knowledge, this is the first work that formulates the phishing detection task as a dynamic graph classification problem. Moreover, the dynamic graph classification can disperse disturbances in multiple time slices when mitigating the impact of malicious transactions, thereby enhancing the robustness of the detector.

2) We constructed TEGs that model both the structural and temporal information and proposed the TEGDetector to capture rich information for reflecting the user behavior features. Besides, a fast nonparametric phishing detector (FD) is presented benefiting from an insight, that is, phishers have more potential trading partners than normal addresses, which improves the detection performance and efficiency of the phishing detector.

3) Experiments conducted on the Ethereum dataset demonstrate that TEGDetector can achieve SOTA detection performance. Besides, phishing deception experiments indicate that TEGDetector is more robust in phishing detection than the existing methods.

The rest of this article is organized as follows. Related works are introduced in Section II, while the proposed method is detailed in Section III. Experiment results and discussion are showed in Section IV. Finally, we conclude our work.

II. RELATED WORK

In this section, we briefly review the existing works on phishing detection and graph classification.

A. Phishing Detector

To provide early warnings to potential victims, various phishing detectors are proposed to identify phishing addresses. Feature engineering-based phishing detectors [10], [11] usually manually extract basic and additional transaction statistical features from preprocessed transactions, then use them to train a classifier. To realize automatic phishing behavior feature extraction, walking-based phishing detectors learn the user’s transaction behavior feature unsupervised. Wu et al. [14] performed a biased walking according to the transaction amount and timestamp, then obtained the address sequence to extract the user’s behavior features. Lin et al. [15] further defined the temporal weighted multidigraph, which ensures that the walking sequences contain the actual meaning of the currency flow. The subgraph-based phishing detectors pay more attention to spatial structure information. Yuan et al. [16] designed second-order subgraphs to represent the target address, modeling the phishing detection task as a graph classification problem. Wang et al. [20] mapped the original transaction subgraphs to the more complex edge subgraphs. Shen et al. [18], and Zhang and Chen [19] introduced GNNs to realize blockchain phishing detection in an end-to-end manner. In general, the existing blockchain phishing detectors will sacrifice some structural or temporal information when capturing users’ behavior features. Moreover, the robustness of phishing detectors lacks research.

The above-mentioned methods mainly deal with graph data modalities, while several recent works [28], [29], [30], [31] put their efforts on other data modalities. For instance, Wei et al. [32] presented a way to detect such malicious URL using convolutional neural networks, which can detect zero-day attacks. Mahdavisharif et al. [33] designed a specific architecture of long short-term memory, and it can detect complex relationships and long-term dependencies between incoming traffic packets. Inspired by the empirical findings on feature evaluation, Evans et al. [34] proposed a reinforcement-learning system that can automatically find the optimum features for detecting different types of attacks.

B. Graph Classification

For a blockchain transaction platform, the transactions of the target address and its neighbors are usually sufficient to reflect its transaction pattern. Intuitively, it is possible to model the phishing detection task as a graph classification problem.

There are two general approaches to graph classification. The first [35], [36], [37], [38] assumes that molecules with similar structures share similar functions, and converts the core problem of graph learning to measure the similarity of different graphs. The second [39], [40], [41], [42] introduces various pooling operations to aggregate the node-level representations into the graph level, which performs better on complex graphs. Besides, for the importance of different nodes in a node’s neighborhood, Velickovic et al. [27] proposed to pay attention to those nodes that play a larger role, while ignoring some nodes that play a smaller role, that is, to assign different weights to each node.

It is worth noting that although numerous dynamic graph mining methods [43], [44], [45], [46], [47] have been studied, most graph classifiers are designed for static graphs. Due to the dynamic evolving pattern of user behaviors, a dynamic graph classifier will be beneficial to phishing detection.

III. METHODOLOGY

TEGDetect seeks to identify phishing addresses by extracting their evolving behavior cues from the transaction graphs. An overview of the proposed method is outlined in Fig. 2. In the following, we present the details of each component. For convenience, the definitions and abbreviations of important symbols used are listed in Nomenclature.

A. Problem Formulation

Before the explanation of TEGDetector in detail, we first give some necessary definitions. A graph is represented as $G = (V, E)$, where $V$ is the node set, $E$ is the edge set. $G$ usually contains an attribute vector of each vertex. Here, we denote the attributes of graph $G$ as $X$ and $A \in \{0, 1\}^{n \times n}$ as the adjacency matrix, and use $G = (A, X)$ to represent a graph more concisely. In our work, the phishing detection task is modeled as a dynamic graph classification problem.
**Definition 1 (Transaction Evolution Time):** Given the target transaction set \( E_i = \{e_0, \ldots, e_{|E|}\} \), which represents the neighborhood transaction set extracted with the target address \( v_i \) as the center. For the target address \( v_i \), the transaction evolution time of \( e_j \in E \) is defined as follows:

\[
ET(e_j, v_i, E) = \frac{t_j - t_{min}}{t_{max} - t_{min}}
\]

where \( t_j \) is the timestamp of \( e_j \), \( t_{max} \) and \( t_{min} \) represent the maximum and minimum timestamps in \( E \), respectively.

**Definition 2 (Dynamic Graphs):** Given a dynamic graph with length \( T \), denoted as \( G = \{G_{t-T}, G_{t-T+1}, \ldots, G_{t-1}\} \), where \( G_{t} = (V, E_{t}) \) denotes the \( k \)th time slice of the dynamic graph. The adjacency matrix of \( G_{t} \) is denoted by \( A_{k} \) whose element \( a_{k,i,j} = 1 \) if there is an edge from node \( i \) pointing to node \( j \) on \( k \)th time slice, otherwise \( a_{k,i,j} = 0 \).

**Definition 3 (Phishing Detection on Blockchain):** On blockchain transaction network, the node set \( V \) represents the users of the blockchain trading platform, and \( E \) represents the transaction record set between different users. Here, user \( v_i \) has \( n \) transactions \( E_i = \{e_0, e_1, \ldots, e_n\} \) with other users. To enhance transaction evolution information, we utilize the transaction evolution time to divide \( E_i \) into \( T \) time slices, getting the dynamic graph \( G_i \). Therefore, the blockchain network with \( N \) users can be modeled as a dynamic graph classification dataset \( G_{\text{set}} \), including \( N \) graphs \( \{G_1, G_2, \ldots, G_N\} \). The phishing detection task aims to predict the categories of unlabeled dynamic graphs through the model \( f_\theta(\cdot) \) trained by the labeled dynamic graphs with the corresponding label, that is, normal address and phishing address.

\[
\arg\min_{\theta} \sum_{G_i \in G_{\text{train}}} \mathcal{L}(f_\theta(G_i), y_i) = \arg\min_{\theta} \sum_{G_i \in G_{\text{train}}} \mathcal{L}(f_\theta(G_i), y_i) \tag{2}
\]

where \( G_{\text{train}} \) is the set of train dynamic graph. \( y_i \) denotes the \( i \)th dynamic graph corresponding true label, where \( y_i = 0 \) denotes that the target address is normal address, and \( y_i = 1 \) represents it is phishing address.

**Definition 4 (Deception on Phishing Detection):** For a given target dynamic graph set \( G_{\text{set}} \) and the training dynamic graph set \( G_{\text{train}}, G_i' \) is the target deception dynamic graph generated by slightly perturbing the structure \( A \) or attributes \( X \) of the original target dynamic graph \( G_i \subset G_{\text{set}} \). The deception attack aims to maximize the loss of the target deception dynamic graph \( G_i' \) on \( f_\theta(\cdot) \), causing \( G_i' \) to get the wrong prediction result. Here, we focus on the deception attack on the graph structure, which can be defined as

\[
\max \sum_{G_i' \in \Psi(G_i)} \mathcal{L}(f_\theta(G_i'), y_i) \tag{3}
\]

subject to \( \theta^* = \arg\min_\theta \sum_{G_i \in G_{\text{train}}} \mathcal{L}(f_\theta(G_i), y_i) \)

where \( \Psi(G_i) \) is the target deception dynamic graph set. \( f_\theta(\cdot) \) denotes the target phishing detector, \( y_i \) denotes the \( i \)th dynamic graph corresponding true label.

**B. Data Preprocessing**

The phishing detection problem on the blockchain is a typical supervised learning problem, which requires labeled user addresses to train TEGDetector. Here, we obtained an Ethereum address list from the blockchain academic research data platform Xblock.\(^2\) We extract transaction sending/receiving addresses, transaction amount, timestamps, and address labels as the crucial information for constructing TEGs. The sending addresses and the receiving ones correspond to the nodes on graph, and the transaction amount and timestamps represent the edge weight and temporal information between the node pairs, respectively. Moreover, we construct the address label-based attribute \( X \in \mathbb{R}^{N \times 2} \) for \( N \) addresses. \( X_{i,1} = 1 \) if the address \( v_i \) is a phishing address, and \( X_{i,0} = 1 \) otherwise.

**C. Transaction Evolution Graph Construction**

TEGs utilize graphs to retain the crucial information extracted in Section 3-B. On one hand, constructing a separate subgraph for each target address can preserve its neighborhood structural information as much as possible. On the other hand, the timestamp information reflects the evolution transaction behavior of the address. The TEG with multiple time slices helps frame the evolution process of the target address’s transaction behaviors. Fig. 3 shows the construction process of the TEGs, including target transaction extraction, transaction evolution division, and TEGs construction.

1) **Target Transaction Extraction:** In Fig. 3(a), for each user address \( v_i \in V \), we take it as the central address and select its transaction partners as the first-order transaction addresses. Here, we set the timestamp of the first transaction \( v_0 \) as the start time. Then, we take the same method to reserve these

\(^2\)http://xblock.pro/
first-order addresses’ transaction partners after the start time. For example, as shown on the far left of Fig. 3, there are several transaction records in the table. The transaction record in the red frame corresponds to the red edge “0xb781ce…” in Fig. 3(a), with a weight of 100. We repeat the above steps until \(v_i\) has a \(K\)-order neighbor address set \(N_i\) and a transaction set \(E_i\). Considering that transactions with larger transaction amount tend to better reflect the evolution of the user’s transaction behaviors, we keep the top \(\text{max\_links}\) transactions to further limit the size of the TEGs.

2) **Transaction Evolution Division:** Generally, the original timestamp conveys absolute temporal information. It reflects the overall evolution pattern of the blockchain transaction network, but is not conducive to capture the independent evolution behaviors of the target address. To address this issue, we utilize the transaction evolution time to divide transactions into different time slices, enhancing transaction evolution information.

In Fig. 3(b), we divide \(E_i\) into \(T\) transaction evolution time slices by (1). In this way, we simulated the behavior evolution of \(v_i\) in \(T\) time slices.

3) **TEGs Construction:** We merge multiple transactions between the same addresses into one at the same time slice, taking the summed transaction amount as the new transaction amount. As shown in Fig. 3(c), we finally construct \(T\) directed weighted graphs based on the obtained transaction records as our TEGs. Thus, for each user’s address, we construct TEGs. In this way, each graph in TEGs contains structural information, that is, node and edge information. Utilizing a two-layer graph convolutional network (GCN) module, we map the structural information to a \(d\)-dimensional node representation \(Z\). At the same time, different directed weighted graphs in TEGs include temporal information which can use the idea of GRU to capture. See Section III-D for specific calculation steps.

### D. TEGDetector

In this section, we design a phishing detector for fully extracting the structure and temporal information from TEGs, termed TEGDetector. As shown in Fig. 4, TEGDetector is designed in an end-to-end manner, including EF-extractor, evolution graphs pooling, and behavior recognition.

The EF-extractor integrates structural and temporal information to extract the addresses’ evolution features. Through the alternation with EF-extractor, the evolution graph pooling aggregates the evolution features of similar addresses until obtaining the TEG’s graph-level features. The behavior recognition assigns time coefficients to these graph-level features, and comprehensively considers the target address’s transaction behaviors in different time slices, which also enhances the robustness of TEGDetector.

1) **EF-Extractor:** We introduce EF-extractor to learn the user addresses’ transaction evolution features at different time slices. Since graph convolutional layers have proven their powerful ability to capture the structural features of graphs in [48] and [49], EF-extractor employs graph convolutional layers to learn the structural features of the current TEG slice. Meanwhile, we learn from the idea of GRU [50] to capture the temporal information of the TEGs. Another reason for choosing GRU is that it has fewer model parameters and runs faster than long short-term memory [51].

Specifically, EF-extractor utilizes a two-layer GCN module to map the structural information to a \(d\)-dimensional node representation \(Z\). As the structural features of the \(t\)th slice of TEGs, \(Z_t\) can be defined as

\[
Z_t = \text{GCN}(h_{t-1}, A_t) = f(\hat{A}_t, \sigma(\hat{A}_t h_{t-1} W_0) W_1)
\]

where \(\hat{A}_t = \tilde{D}_t^{-1/2} \tilde{A}_t \tilde{D}_t^{-1/2}\), \(A_t \in \mathbb{R}^{N \times N}\) is the adjacency matrix of the \(t\)th slice of TEGs, \(\tilde{A}_t = A_t + I_{N(t)}\) is the adjacency matrix with self-connections. \(D_{t(i)} = \sum_j A_{t(ij)}\) denotes the degree matrices of \(\hat{A}_t\), \(h_{t-1}\) is the evolution features of the \(t\)th slice, which will be described in detail later. \(W_0 \in \mathbb{R}^{N \times H}\) and \(W_1 \in \mathbb{R}^{H \times d}\) denote the weight matrix of the hidden layer and the output layer, respectively. \(\sigma\) is the Relu active function and the input \(h_0 = X\).

For the evolution process of the structural features, EF-extractor first calculates the update gate \(z_t\) and the reset gate \(r_t\) according to the current structural features \(Z_t\) and the previous evolution features \(h_{t-1}\), which can be expressed as

\[
z_t = \sigma(Z_t W_z + h_{t-1} U_z)
\]

\[
r_t = \sigma(Z_t W_r + h_{t-1} U_r)
\]

where \(W_z, W_r \in \mathbb{R}^{N \times d}\) and \(U_z, U_r \in \mathbb{R}^{d \times d}\) are the weight matrix of the update/reset gate, respectively. These parameters are responsible for different functions. \(W_z, W_r\) control the degree to which the state information of the previous time slices is brought into the current state. \(W_z, W_r\) control how much information of the previous state is written to the candidate hidden state. Their parameter values are randomly generated before the model training. The update gate decides how much \(h_{t-1}\) is passed to the future, and the reset gate determines how much \(h_{t-1}\) need to be forgotten.

The next step is to calculate the candidate hidden state \(\hat{h}_t\) by reset gate. Here, EF-extractor stores historical evolution features \(h_{t-1}\) and memorizes the current state

\[
\hat{h}_t = \tanh(W Z_t + (r_t \odot h_{t-1}) U)
\]

where \(W \in \mathbb{R}^{N \times d}\) and \(U \in \mathbb{R}^{d \times d}\) are the weight matrix used to calculate \(\hat{h}_t\) and denotes the Hadamard product.

Finally, EF-extractor updates the current evolution features \(h_t\) according to \(h_{t-1}\) and \(\hat{h}_t\)

\[
h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t.
\]

2) **Evolution Graphs Pooling:** Intuitively, addresses with similar evolution features can be divided into the same address clusters. This motivates us to aggregate similar addresses...
Equations (10) and (11) generate the evolution features \( v_i \) addressing \( \tilde{d} \) malicious transactions. Therefore, we comprehensively consider the evolution features in all slices rather than using only the most recent time slice, which can alleviate the negative impact of malicious transactions.

After extracting the evolution features \( \{h_i^{\text{pool}}, \ldots, h_T^{\text{pool}}\} \) for \( T \) time slices (\( T = 1, h_i \in \mathbb{R}^{1 \times d}, t \in [1, \ldots, T] \)), the Read-out operation assigns time coefficients \( \alpha = [\alpha_1, \ldots, \alpha_T] \) to different evolution features, aggregating the \( T \) evolution features into the unique evolution feature \( h_i \) for the target address \( v_i \)

\[
h_i = \sum_{t=1}^{T} \alpha_t h_i^{\text{pool}}
\]

where \( h_i^{\text{pool}} \) denotes \( v_i \)'s evolution features of \( t \)th slice.

Finally, we take \( h_i \) as the input of the MLP layer with a softmax classifier. Moreover, we use the cross-entropy function \( \mathcal{L} \) to train TEGDetector, which is given by

\[
\hat{Y} = \text{softmax}(\text{MLP}(h_i))
\]

\[
\mathcal{L} = - \sum_{G_i \in \mathcal{G}_{\text{set}}} \sum_{j=1}^{|\tau|} Q_{ij} \ln \hat{Y}_{ij}(A', X')
\]

where \( G_i \in \mathcal{G}_{\text{set}} \) denotes the target address \( v_i \)'s TEG in the training set \( \mathcal{G}_{\text{set}} \), \( Y = \{\tau_1, \ldots, \tau_n\} \) is the category set of the TEGs, \( Q_{ij} = 1 \) if \( G_i \) belongs to category \( \tau_i \) and \( Q_{ij} = 0 \) otherwise. \( \hat{Y}_{ij} \) denotes the predicted probability of \( G_i \), which is calculated by (13) and can be considered as a function of \( A' \) and \( X' \), thus we denote it as \( \hat{Y}_{ij}(A', X') \).

### E. Time Complexity Theoretical Analysis

The time cost of TEGDetector mainly comes from three parts, including the time cost for extracting the addresses’ evolution features \( (T_{\text{extr}}) \), the time cost to aggregate similar addresses \( (T_{\text{agg}}) \), and the time cost of transactional behavior recognition \( (T_{\text{rckn}}) \). Therefore, the time complexity of TEGDetector is

\[
O(T_{\text{extr}}) + O(T_{\text{agg}}) + O(T_{\text{rckn}}) \sim O(n)
\]

where \( T_{\text{extr}} \) is contingent on both the number of testing examples and the scale of the graph. \( T_{\text{agg}} \) depends on the budget of the extracted evolution features, and \( T_{\text{rckn}} \) is the complexity of the transactional behavior recognition. Therefore, based on all the aforementioned steps, \( O(n) \) indicates that the time complexity of TEGDetector is linear.

### IV. EXPERIMENTS

In this section, we comprehensively evaluate the proposed TEGDetector, including its phishing detection performance, detection efficiency, and robustness.

#### A. Experimental Settings

1) Model Structure: In our TEGDetector, we adopt two EF-extractor modules, and each one is followed by an evolution graph pooling operation. Particularly, in the behavior recognition operation, the MLP consists of two fully connected layers, followed by a softmax classifier.
2) Parameter Settings: We introduce the parameter settings for the best detection performance in the experiment. In the construction of TEGs, we set the maximum neighbor order \( K \) to 2. The number of address clusters for the two evolution graph pooling operations are \( \hat{N}_1 = N \ast 0.1 \) and \( \hat{N}_2 = 1 \). The max_links and the node representation dimension \( d \) are set to 2k and 64, respectively. The number of neurons in each MLP layer is set to 64 and 2. We use the Adam optimizer to optimize the model and search for the learning rate in \((0.1, 0.01, 0.001)\).

3) Metric Settings: Since the addresses have only two categories, we evaluate the detection performance by precision, recall, and F1-score. When verifying the robustness of different detectors, we only conduct experiments on phishing addresses, so the accuracy can be used to evaluate the robustness of phishing detectors.

B. Datasets

We evaluate TEGDetector on the real-world Ethereum transaction dataset released on the Xblock platform.\(^3\) They collected Ethereum transaction data from the Ethereum trading platform\(^4\) through Ethereum clients Geth and Parity. Each transaction data in this website contains dozens of attributes, among which the transaction timestamp, transaction sending and receiving addresses, and transaction amount are the key information for constructing a transaction network. The sending address and receiving address correspond to the nodes in the transaction network. The transaction timestamp and the transaction amount indicate the edges between the corresponding node pair. Xblock collects the current mainstream blockchain data and is one of the blockchain data platforms in the academic community. All blockchain datasets have been cleaned and classified in a standard way, which can be easily downloaded in a standard and consistent format.

In the real Ethereum network, there will be far more normal addresses than phishing addresses. Due to the imbalanced distribution, the model will be underfitted if we directly use the unprocessed data for training. Several methods [24], [25], [26] have been proposed for imbalanced data learning, they integrate the sampling idea and cost-learning into an efficient end-to-end deep learning framework. Inspired by them, we under-sample the normal addresses to adjust the sample distribution, which is beneficial to the training of the model.

Xblock provides 1660 phishing addresses that have been reported and 1700 randomly selected normal ones with the records of their two-order transactions. Specifically, we randomly selected 1000 phishing addresses and the same number of ordinary addresses and construct the TEGs with ten time slices (\( T = 10 \)) for them. To make a comprehensive evaluation, we divide the TEGs into two parts: \( \{60\%, 70\%, 80\%\} \) as the training set and the remaining \( \{40\%, 30\%, 20\%\} \) as the test set. The basic statistics are summarized in Table I. In the experiment, we repeated the above steps five times and reported the average phishing detection performance. The compared methods are briefly described as follows.

Density detector calculates the density ratio of TEG slices containing transactions to all slices. When the density ratio is greater than 0.5, the target address will be classified as a phishing address. Repeat detector calculates the repetition ratio of test transactions with the same direction as the training ones to all test transactions. According to the conclusion of Lin et al. [52], we classify addresses with a repetition ratio greater than 0.1 as phishing addresses. Note that in this case, we divide the slices in each TEG into training slices and test ones according to different division ratios. Deepwalk [12] and Node2vec [13] learn node representations through random walking and can be used in blockchain transaction networks. Trans2vec [14] and T-EDGE [15] consider the transaction amount and timestamps of blockchain transactions on the basis of random walking, thus achieving better phishing detection performance. PBGNN [18] and MCGC [19] are graph classifiers designed for phishing detection on the blockchain. They have achieved satisfactory detection performance and are easy to implement for phishing detection on new addresses. Transaction subgraph network (TSGN) [20] mapped the original transaction subgraphs to the more complex edge subgraphs. In addition, Diffpool [41] aggregates the nodes into a new cluster as the input of the next layer by utilizing a cluster assignment matrix. SAGPool [42] selects top-\( K \) nodes based on a self-attention mechanism to form the induced subgraph for the next input layer. We extend them for phishing detection in this work since they have achieved encouraging results on graph classification tasks.

D. Performance of TEGDetector

In this section, we discuss the phishing detection performance of TEGDetector, and analyze its detection efficiency.

1) Phishing Detection Performance: Compared with other detectors in Table II, TEGDetector achieves SOTA performance at different training set ratios. Specifically, although the density detector achieves a recall of 99.30\%, a precision of 50.41\% indicates that it is actually invalid to detect phishing addresses based on density ratio. Compared with density detector and repeat detector, other compared methods achieve better detection performance with automated feature learning. Unfortunately, the lack of structural or temporal

| TEG properties | Dataset Statistics |
|----------------|--------------------|
| # Addresses    | #Transactions      | Average degree |
| Sum            | 790,849            | 3,383,022       |                          |
| Average        | 395.42             | 1,691.31        | 4.86                      |
| Maximum        | 4,934              | 110,060         | 7.09                      |
| Minimum        | 2                  | 1                | 1.00                      |

3https://etherscan.io/

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
TABLE II
DETECTION PERFORMANCE OF DIFFERENT PHISHING DETECTORS. WE USE BOLD TO HIGHLIGHT WINS

| Compared methods                  | Training ratio |
|-----------------------------------|----------------|
|                                   | 60%            | 70%            | 80%            |
|                                   | Precision | Recall | F1-score | Precision | Recall | F1-score | Precision | Recall | F1-score |
| Density detector                  | 50.41     | 99.30  | 66.87    | 50.41     | 99.30  | 66.87    | 50.41     | 99.30  | 66.87    |
| Repeat detector                   | 46.94     | 51.78  | 49.25    | 48.16     | 62.35  | 54.35    | 48.53     | 69.62  | 57.19    |
| Deepwalk [12]                     | 76.85     | 68.85  | 72.63    | 77.90     | 71.20  | 74.40    | 78.15     | 72.70  | 75.33    |
| Node2vec [13]                     | 81.65     | 70.35  | 75.58    | 82.30     | 72.20  | 76.92    | 82.65     | 74.85  | 78.56    |
| Trans2vec [14]                    | 78.65     | 77.90  | 78.27    | 88.65     | 86.55  | 87.59    | 91.45     | 87.65  | 89.51    |
| T-EDGE [15]                       | 79.05     | 76.20  | 77.60    | 87.45     | 76.65  | 81.69    | 88.75     | 78.55  | 83.34    |
| F^BGNN [18]                       | 88.65     | 91.45  | 90.03    | 89.20     | 91.55  | 90.36    | 89.20     | 92.05  | 90.60    |
| MCGC [19]                         | 90.50     | 91.55  | 91.02    | 90.55     | 92.10  | 91.32    | 90.75     | 92.85  | 91.79    |
| Diffpool [41]                     | 89.34     | 89.05  | 89.19    | 89.73     | 89.46  | 89.59    | 92.84     | 92.87  | 92.85    |
| SAGPool [42]                      | 84.70     | 84.36  | 84.53    | 87.20     | 86.65  | 86.92    | 91.54     | 90.86  | 91.20    |
| TSGN [20]                         | 86.06     | 83.59  | 84.25    | 83.54     | 82.59  | 83.01    | 82.29     | 83.19  | 82.51    |
| TEGDetector (Ours)                | **95.90** | **95.60** | **95.75** | **96.55** | **96.75** | **96.65** | **96.30** | **96.25** | **96.28** |

information restricts them from achieving more accurate phishing detection. In contrast to Trans2vec and T-EDGE (which focus on temporal information), F^BGNN, MCGC, and TSGN (which pay attention to structural information), TEGDetector achieves the best detection performance. Besides, TEGDetector improves the precision by 6.16% and 10.80% compared to Diffpool and SAGPool at 60% training set ratios. This suggests that balancing structural and temporal information can more accurately capture the transaction behaviors of the target address. Besides, we can observe that most of all methods are getting better and better as the training ratio increases. This is reasonable because as the training ratio increases, the learned features are more abundant, the better the model fitting effect is, the better the inference effect can be achieved. But TEGDetector has a situation of higher precision of 70% than 60%, but lower precision than 80%. It is not surprising. That’s because we noticed that the method has achieved very high accuracy when the training ratio is 60%, that is, the effect gradually improves to the best.

Fig. 5(a) shows the time coefficients learned by TEGDetector, pointing out that the earliest and recent transactions contribute more than other periods. Intuitively, they reflect the purpose of creating the address and the latest transaction behaviors, respectively. To further explore the significance of the learning time coefficient, we use the learned time coefficient to perform a weighted summation for multiple transactions between the same addresses when constructing the static subgraph, termed MCGC_T. In Fig. 5(b), the detection performance of MCGC has been further improved. We can conclude that considering the time coefficients can indeed improve the phishing detector’s performance.

2) Detection Efficiency: We further study the detection efficiency of TEGDetector. Since TEGs are essentially a series of dynamic subgraphs, we compare TEGDetector with the GNN-based phishing detectors which input the static ones.

Fig. 6(a) and (b) show the training time and detection time of different detectors. We select all training addresses for model training and count the detection time of 100 randomly selected addresses in the detection phase. We can observe that with the increase of max_links, the training time of TEGDetector becomes longer than that of F^BGNN. Moreover, TEGDetector’s detection time still reaches 6.5 times that of F^BGNN, although the gap has been greatly reduced. Considering TEGDetector’s excellent detection performance, we believe that such a price is acceptable.

E. Fast and Nonparametric Phishing Detection

The previous section confirms the SOTA phishing detection performance of TEGDetector. However, the high time complexity of TEGDetector is still a challenge. To address this problem, we propose to quickly filter out the obvious normal addresses while ensuring that the real phishing ones are not missed, which can narrow down the search space of suspicious addresses.

Phishers usually send phishing messages to massive users, allowing them to have more potential transaction partners. We believe that they may have more intensive large-amount transactions compared to the normal addresses. To verify our conjecture, we defined central transaction ratio (CTR), which...
Fig. 7. CTR distribution of the normal/phishing addresses. We select the top \( k \) (20\%, 40\%, 60\%) transactions as our target based on the transaction amount. In the TEGs centered on phishing addresses, large-amount transactions are more concentrated in the central address. (a) Top 20\%. (b) Top 40\%. (c) Top 60\%.

Fig. 8. (a) FD's detection performance under different CTR thresholds. (b) FD can improve the detection performance of TEGDetector and reduce its detection time.

represents the ratio of the central address’s transactions to all transactions in a TEG (or a static subgraph). Fig. 7 shows that phishing addresses are more likely to have a CTR higher than 0.9 at the same \( k \) (top \( k \) transactions based on the transaction amount). As \( k \) decreases, large-amount transactions tend to be more concentrated in the central address, which is more obvious in phishing addresses.

Inspired by this observation, we propose a fast and non-parametric detector. Specifically, we classify the target address whose CTR is greater than a threshold as a phishing address, otherwise, it is regarded as a normal address. In Fig. 8(a), when CTR is set to 0.6, FD can almost reach 100\% recall, and precision almost reaches the highest 60\%. This indicates that FD can filter out normal addresses almost without missing any phishing addresses, which can be a predetermined approach before using TEGDetector for precise phishing detection. Meanwhile, Fig. 8(b) shows that the TEGDetector’s precision improved more than its recall since FD may prefilter some normal addresses that may be misclassified by TEGDetector. Additionally, we can observe that FD can also reduce the detection time of TEGDetector by approximately 15\%. Consequently, FD is a light-weighted solution for phishing detection when we consider both performance and efficiency.

F. Ablation Study of TEGDetector

To further explore the effectiveness of TEGDetector, we conduct ablation experiments on pooling layer, time coefficient, and GRU module. We utilize the average pooling and maximum pooling operations on the feature matrix of the graph, respectively, expressed as TEGD-ave and TEGD-max. For the time coefficients, we replace the weighting of time coefficients in TEGDetector with a summation operation to obtain the variant TEGDetector_S. To explore the influence of the GRU module for capturing these historical information features in TEGDetector, we conduct ablation study on the GRU module in EF-extractor. We denote the TEGDetector without the GRU module as TEGD_Nodyn.

As illustrated in Table III, the performance of TEGDetector is better than the two detection methods with average and maximum pooling, for example, the precision of TEGDetector is 96.55\%, while the precision of TEGD-max is 95.24\%. This indicates that the pooling method using the cluster assignment matrix can extract graph-level features more effectively than the average and maximum pooling methods. We also observe that TEGDetector_S is almost 2\% lower than TEGDetector on precision, recall, and F1-score, which demonstrates that time coefficients give different weights on different timestamps to improve model performance. Empirically, TEGDetector surpassed TEGD_Nodyn by 4.66\%, 5.83\%, and 5.25\% in terms of precision, recall, and F1-score, respectively. The strong empirical evidences suggest that the GRU module in TEGDetector can well capture the dynamic evolution features in the graphs.

G. Robustness of TEGDetector

Now we evaluate the robustness of detectors when phishers maliciously conceal their phishing behaviors. From the perspective of network topology properties and detectors, we design two methods to add perturbations to the transaction networks, that is, the CTR-based and gradient-based methods. According to our discussion in the previous section, the phishing addresses' CTRs are more likely to be larger than 0.9. Therefore, we designed a phishing deception experiment where we change the phishing address’s CTR. As shown in Fig. 9(a), we randomly add transactions to the noncentral address pairs in TEGs until their CTRs are less than the set value. Specifically, we randomly select \( T/2 \) time slices in TEGs and add malicious transactions to them. The transaction amount of these transactions is set to a random value less than the maximum one in original TEGs.

In the phishing deception experiment, the smaller the target CTR, the more malicious transactions need to be added. In Fig. 9(b), the detection accuracy of Trans2vec and T-EDGE
is more stable than other existing methods. We speculate that although the randomness of the walking strategies leads to the loss of structural information, it also reduces the impact of malicious transactions. In contrast, the static subgraphs constructed by I$^2$BGNN and MCGC retain all the malicious information, their detection accuracy reduces drastically. This further demonstrates that in addition to limiting the phishing detection performance, the robustness of static subgraphs lacking temporal information is also worrisome. Reassuringly, TEGDetector still with an accuracy of 83% even in the worst case (when the target CTR is set to 0.2), indicating that it is necessary to expand the static subgraphs into the dynamic ones. Compared with existing methods, TEGDetector can fully balance the target address’s transaction behaviors in all periods, which enables it to capture more comprehensive behavior features. It is worth noting that TEGDetector is more robust than TEGDetector_S, that is, the latter undergoes an accuracy decline of 21.25%, while the former with only a decline of 13.5%. This testifies that TEGDetector is significantly more robust when phishers cannot add malicious transactions in each period.

In addition to considering the malicious transactions generated based on the network topology properties, we also design the malicious transactions generated based on the feedback of the detectors’ gradient. Since T-EDGE and Trans2vec are based on random walk methods, these two detection methods do not perform gradient attacks and compare with other detection methods. As shown in Fig. 10, we obtain the gradient value of the transaction network adjacency matrix from the objective loss function in descending order and add transactions in the order where there are no transactions. The value obtained by multiplying the modify rate of link by the maximum number of nodes in the transaction network is the number of malicious transactions added.

As shown in Fig. 10(a), TEGDetector exceeds 85% in precision, recall, and F1-score even if the modify rate of link is 0.5, which is better than the other three detection methods, for example, precision on the TEGDetector_S, the recall on MCGC, and the F1-score on I$^2$BGNN are only 86.09%, 79.33%, and 72.50%, respectively, in the worst case. From the overall decline in precision, the maximum rate of decrease in

precision of TEGDetector is only 9.02% and TEGDetector_S is 8.91%, while MCGC and I$^2$BGNN are 10.08% and 11.41%, respectively. This indicates that the TEGs can disperse disturbances in multiple timestamps when mitigating the impact of malicious transactions, thereby enhancing the robustness of the detector.

H. TEGDetector on Conventional Datasets

TEGDetector is a framework for handling the dynamic graph classification task modeled by the blockchain phishing detection problem. It is not limited to processing blockchain transactions, but also can be applied to conventional datasets. To verify the effectiveness of TEGDetector, we supplemented two conventional dynamic datasets, that is, DBLP-3 and Reddit datasets. The DBLP-3 dataset is extracted from DBLP (https://dblp.org/), which provides a large collection of bibliographic information from major conferences and journals in various fields. The Reddit dataset is generated from Reddit (https://www.reddit.com/), a social news aggregation, web content rating, and discussion website. The basic information of the network is shown in Table IV. TEGDetector and two baseline methods with the best performance, that is, I$^2$BGNN and MCGC, conduct evaluation experiments on the two conventional datasets.

As shown in Table V, TEGDetector realizes the superior classification performance on conventional datasets than I$^2$BGNN and MCGC. For instance, the precision of TEGDetector is 73.69% on DBLP-3, whereas those of I$^2$BGNN and MCGC are 68.35% and 68.52%, respectively. In addition, for the Reddit dataset, TEGDetector improves the F1-score by 4.31% and 3.40% compared to I$^2$BGNN and MCGC, respectively. This further verifies that TEGDetector can better
extract the latent features of the network to achieve better performance.

I. Visualization of Transaction Evolution Graphs

In this section, we study the availability of the proposed dynamic graph method, that is, whether this method can obtain a dynamic TEG which model both the structural and temporal information to benefit the subsequent phishing detection algorithms. Taking the real-world Ethereum transaction dataset as input, we construct a separate subgraph for each target address in tabular form and use Gephi\(^4\) to select several moments for visualization.

In Fig. 11, we plot the evolutionary trend of 2000 addresses including normal and phishing addresses at different moments. Since we focus on structural features and time features, what we show is the TEG over time for a certain target address. As shown in Fig. 11(a)–(e), we can observe that the transaction graph has been evolving from moment \(t_0\) to moment \(t_1\). Compared to static graphs, our method is better at capturing changes due to temporal features to prove the availability of the proposed dynamic graph method.

V. CONCLUSION

We model phishing detection as a dynamic graph classification problem through constructing the TEGs that can frame both structural and temporal behavior cues. Furthermore, we proposed TEGDetector, a dynamic graph classifier suitable for identifying the target address’s transaction behavior from TEGs. Abundant experimental results demonstrate that TEGDetector achieves almost 7% performance improvement compared to current SOTA. Moreover, we gain insights that in the TEGs, large-amount transactions tend to be more concentrated on phishing addresses. Inspired by this, a fast phishing detector is designed, which can quickly narrow down the search space of suspicious addresses and improve the
detection efficiency of the phishing detector. In possible phishing deception experiment, TEGDetector shows significantly higher robustness than other phishing detectors.

A limitation of TEGDetector lies in its higher time complexity, which stems from the fact that it both considers structural and temporal information. For future work, we plan to explore a lower-complexity phishing detector. In addition, improving the robustness of phishing detectors against more targeted phishing deception methods deserve further research.

REFERENCES

[1] M. Iansiti and K. R. Lakhanl, “The truth about blockchain,” Harvard Bus. Rev., vol. 95, no. 1, pp. 118–127, 2017.
[2] G. Wood, “Ethereum: A secure decentralised generalised transaction ledger,” Ethereum Project Yellow Paper, vol. 151, pp. 1–32, Apr. 2014.
[3] Y. Yuan and F.-Y. Wang, “Blockchain and cryptocurrencies: Model, techniques, and applications,” IEEE Trans. Syst., Man, Cybern., Syst., vol. 48, no. 9, pp. 1421–1428, Sep. 2018.
[4] M. Vasek and T. Moore, “There’s no free lunch, even using Bitcoin: Tracking the popularity and profits of virtual currency scams,” in Proc. Int. Conf. Financial Cryptog. Data Secur. Berlin, Germany: Springer, 2015, pp. 44–61.
[5] W. Chen, Z. Zheng, J. Cui, E. Ngui, P. Zheng, and Y. Zhou, “Detecting Ponzi schemes on Ethereum: Towards healthier blockchain technology,” in Proc. World Wide Web Conf., 2018, pp. 1409–1418.
[6] M. Vasek and T. Moore, “Analyzing the Bitcoin Ponzi scheme ecosystem,” in Proc. Int. Conf. Financial Cryptog. Data Secur. Berlin, Germany: Springer, 2018, pp. 101–112.
[7] Y. Huang et al., “Understanding (mis)behavior on the EOSIO blockchain,” Proc. ACM Meas. Anal. Comput. Syst., vol. 4, no. 2, pp. 1–28, 2020.
[8] S. Hu, Z. Zhang, S. Lu, B. He, and Z. Li, “Sequence-based target coin prediction for crytocurrency pump-and-dump,” Proc. ACM Manage. Data, vol. 1, no. 1, pp. 1–19, May 2023.
[9] EtherScamDB. (2018). EtherScamDB. [Online]. Available: https://blog.chainalysis.com/reports/ethereum-scams
[10] Y. Li, Y. Cai, H. Tian, G. Xue, and Z. Zheng, “Identifying illicit addresses in Bitcoin network,” in Proc. Int. Conf. Blockchain Trustworthy Syst. Berlin, Germany: Springer, 2020, pp. 99–111.
[11] Y.-J. Lin, P.-W. Wu, C.-H. Hsu, I.-P. Tu, and S.-W. Liao, “An evaluation of Bitcoin address classification based on transaction history summarization,” in Proc. IEEE Int. Conf. Blockchain Cryptocurr. (ICBC), May 2019, pp. 302–310.
[12] B. Perozzi, R. Al-Rfou, and S. Skiena, “DeepWalk: Online learning of social representations,” in Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2014, pp. 701–710.
[13] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2016, pp. 855–864.
[14] J. Wu et al., “Who are the phishing? Scam phising detection on Ethereum via network embedding,” IEEE Trans. Syst., Man, Cybern., Syst., vol. 52, no. 2, pp. 1156–1166, Feb. 2022.
[15] D. Lin, J. Wu, Q. Yuan, and Z. Zheng, “T-EDGE: Temporal weighted multidigraph embedding for Ethereum transaction network analysis,” Frontiers Phys., vol. 8, pp. 204, Jun. 2020.
[16] Z. Yuan, Q. Yuan, and J. Wu, “Phishing detection on Ethereum via learning representation of transaction subgraphs,” in Proc. Int. Conf. Blockchain Trustworthy Syst. Berlin, Germany: Springer, 2020, pp. 178–191.
[17] Y. Wang, Z. Liu, J. Xu, and W. Yan, “Heterogeneous network representation learning approach for Ethereum identity identification,” IEEE Trans. Computat. Social Syst., vol. 10, no. 3, pp. 890–899, Jun. 2023.
[18] J. Shen, J. Zhou, Y. Xie, S. Yu, and Q. Xuan, “Identity inference on blockchain using graph neural network,” 2021, arXiv:2104.06559.
[19] D. Zhang and J. Chen, “Blockchain phishing scam detection via multichannel graph classification,” 2021, arXiv:2108.08456.
[20] J. Wang et al., “TSGN: Transaction subgraph networks assisting phishing detection in Ethereum,” 2022, arXiv:2208.12938.
[21] D. Zügner, A. Akbarnejad, and S. Günemann, “Adversarial attacks on neural networks for graph data,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2018, pp. 2847–2856.

\(^4\)Gephi is a data visualization processing software in the field of network analysis.
[22] H. Dai et al., “Adversarial attack on graph structured data,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 1115–1124.

[23] H. Chang et al., “A restricted black-box adversarial framework towards attacking graph embedding models,” in Proc. AAAI Conf. Artif. Intell., 2020, vol. 34, no. 4, pp. 3389–3396.

[24] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, “Learning and transferring mid-level image representations using convolutional neural networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Columbus, OH, USA: IEEE Computer Society, Jun. 2014, pp. 1717–1724.

[25] J. Zhang, X. Wu, and V. S. Shengs, “Active learning with imbalanced multiple noisy labeling,” IEEE Trans. Cybern., vol. 45, no. 5, pp. 1095–1107, May 2015.

[26] H. Yu, C. Sun, X. Yang, W. Yang, J. Shen, and Y. Qi, “ODOC-ELM: Optimal decision outputs compensation-based extreme learning machine for classifying imbalanced data,” Knowl.-Based Syst., vol. 92, pp. 55–70, Jun. 2016.

[27] P. Velickovic et al., “Graph attention networks,” Stat., vol. 1050, no. 20, pp. 48510–48550, 2017.

[28] K. Siddique, Z. Akhter, M. A. Khan, Y. Jung, and Y. Kim, “Developing an intrusion detection framework for high-speed big data networks: A comprehensive approach,” KSII Trans. Internet Inf. Syst., vol. 12, no. 8, pp. 4021–4037, 2018.

[29] M. Zaminkar and R. Fotohi, “SoS-RPL: Securing Internet of Things against sinkhole attack using RPL protocol-based node rating and ranking mechanism,” Wireless Pers. Commun., vol. 114, no. 2, pp. 1287–1312, Sep. 2020.

[30] R. Fotohi and H. Pakdel, “A lightweight and scalable physical layer attack detection mechanism for the Internet of Things (IoT) using hybrid security scheme,” Wireless Pers. Commun., vol. 119, no. 4, pp. 3089–3106, Aug. 2021.

[31] R. Fotohi and F. S. Aliee, “Securing communication between things using blockchain technology based on authentication and SHA-256 to improving scalability in large-scale IoT,” Comput. Netw., vol. 197, Oct. 2021, Art. no. 108331.

[32] W. Wei, Q. Ke, J. Nowak, M. Korytkowski, R. Scherer, and M. Woźniak, “Accurate and fast URL phishing detector: A convolutional neural network approach,” Comput. Netw., vol. 178, Sep. 2020, Art. no. 107275.

[33] M. Mahdavisharif, S. Jamali, and R. Fotohi. “Big data-aware intrusion detection system in communication networks: A deep learning approach,” J. Grid Comput., vol. 19, no. 4, p. 46, Dec. 2021.

[34] K. Evans, A. Abuaedba, M. Ahmed, T. Wu, M. Johnstone, and S. Nepal, “RAIDER: Reinforcement-aided spear phishing detector,” CoRR, vol. abs/2105.07582, pp. 1–17, May 2021.

[35] K. M. Borgwardt and H.-P. Kriegel, “Shortest-path kernels on graphs,” in Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2005, p. 8.

[36] N. Sharma, M. A. Khan, Y. Jung, and Y. Kim, “Accurate and fast URL phishing detector: A convolutional neural network approach,” Comput. Netw., vol. 178, Sep. 2020, Art. no. 107275.

[37] M. Mahdavisharif, S. Jamali, and R. Fotohi. “Big data-aware intrusion detection system in communication networks: A deep learning approach,” J. Grid Comput., vol. 19, no. 4, p. 46, Dec. 2021.

[38] K. M. Borgwardt and H.-P. Kriegel, “Shortest-path kernels on graphs,” in Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2005, p. 8.

[39] N. Sharma, M. A. Khan, Y. Jung, and Y. Kim, “Accurate and fast URL phishing detector: A convolutional neural network approach,” Comput. Netw., vol. 178, Sep. 2020, Art. no. 107275.

[40] M. Mahdavisharif, S. Jamali, and R. Fotohi. “Big data-aware intrusion detection system in communication networks: A deep learning approach,” J. Grid Comput., vol. 19, no. 4, p. 46, Dec. 2021.

[41] K. M. Borgwardt and H.-P. Kriegel, “Shortest-path kernels on graphs,” in Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2005, p. 8.

[42] N. Sharma, M. A. Khan, Y. Jung, and Y. Kim, “Accurate and fast URL phishing detector: A convolutional neural network approach,” Comput. Netw., vol. 178, Sep. 2020, Art. no. 107275.

[43] J. Chang, Ç. Gülcüre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” CoRR, vol. abs/1412.3555, pp. 1–9, Dec. 2014.

[44] P. Goyal and E. Ferrara, “Graph embedding techniques, applications, and performance: A survey,” Knowl.-Based Syst., vol. 151, pp. 78–94, Jul. 2018.

[45] T. Li, J. Zhang, P. S. Yu, Y. Zhang, and Y. Yan, “Deep dynamic network embedding for link prediction,” IEEE Access, vol. 6, pp. 29219–29230, 2018.

[46] J. Chen et al., “E-LSTM-D: A deep learning framework for dynamic network link prediction,” IEEE Trans. Syst., Man, Cybern., Syst., vol. 51, no. 6, pp. 3699–3712, Jun. 2021.

[47] J. Chen et al., “Generative dynamic link prediction,” Chaos, Interdiscipl. Nonlinear Sci., vol. 29, no. 12, Dec. 2019, Art. no. 123111.

[48] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in Proc. 5th Int. Conf. Learn. Represent. (ICLR), Toulon, France, Apr. 2017, pp. 1–14.

[49] J. Liu, C. Xu, C. Yin, W. Wu, and Y. Song, “K-core based temporal graph convolutional network for dynamic graphs,” CoRR, vol. abs/2003.09902, pp. 1–13, Mar. 2020.

[50] K. Cho et al., “Learning phrase representations using RNN encoder–decoder for statistical machine translation,” in Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP), A. Moschitti, B. Pang, and W. Daelemans, Eds, Doha, Qatar: ACL, Oct. 2014, pp. 1724–1734.

[51] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[52] D. Lin, J. Chen, J. Wu, and Z. Zheng, “Evolution of Ethereum transaction relationships: Toward understanding global driving factors from microscopic patterns,” IEEE Trans. Computat. Social Syst., vol. 9, no. 2, pp. 559–570, Apr. 2022.

Haibin Zheng received the B.S. and Ph.D. degrees from the Zhejiang University of Technology, Hangzhou, China, in 2017 and 2022, respectively. He is currently a special-term Associate Professor with the Institute of Cyberspace Security, Zhejiang University of Technology. His research interests include deep learning and artificial intelligence security.

Mingya Ma is currently pursuing the master’s degree with the College of Information Engineering, Zhejiang University of Technology, Hangzhou, China. Her research interests include graph data mining, federated learning, and artificial intelligence.

Haonan Ma is currently pursuing the master’s degree with the College of Information Engineering, Zhejiang University of Technology, Hangzhou, China. His research interests include graph data mining, federated learning, and artificial intelligence.
Jinyin Chen (Member, IEEE) received the B.S. and Ph.D. degrees from the Zhejiang University of Technology, Hangzhou, China, in 2004 and 2009, respectively. She studied evolutionary computing at the Ashikaga Institute of Technology, Ashikaga, Japan, in 2005 and 2006. She is currently a Professor with the Zhejiang University of Technology. Her research interests include artificial intelligence security, graph data mining, and evolutionary computing.

Haiyang Xiong is currently pursuing the master’s degree with the College of Information Engineering, Zhejiang University of Technology, Hangzhou, China. His research interests include graph data mining and applications, and artificial intelligence.

Zhijun Yang received the B.Eng. degree from the Huazhong University of Science and Technology, Wuhan, China, in 1989, the M.Sc. degree from Nanjing University, Nanjing, China, in 1992, and the Ph.D. degree from the Federal University of Rio de Janeiro, Rio de Janeiro, Brazil, in 1999. Since 2001, he has been a Post-Doctoral Research Fellow with Heriot-Watt University, Edinburgh, U.K., and then a Senior Research Fellow with The University of Edinburgh, Edinburgh, and the University of Stirling, Stirling, U.K., for ten years. Since 2013, he has been with Middlesex University London, London, U.K., as a Senior Lecturer. Since 2022, he has been with Jack Technology Company, Ltd., Taizhou, China, as a Chief AI Scientist and the Director of the Jack Central Research and Development Institute. His research interests include neural networks, machine learning, system-on-chip design, and neuromorphic engineering.