

PHISHING FRAUD IDENTITY INFERENCE BASED ON GRAPH GATED RECURRENT NEURAL NETWORK

Zhaohuang Chen^{1,†}, Zhongqi Fu^{1,†}, Tao Liang², Haidong Ma¹, Yanfeng Sun¹, Xiaohu Shi^{1,3,*}

¹*College of Computer Science and Technology, Jilin University
Changchun, 130012, China*

²*College of Software, Jilin University
Changchun, 130012, China*

³*School of Big Data and Artificial Intelligence,
Guangdong University of Finance and Economics,
Guangzhou, 510320, China*

*E-mail: shixh@jlu.edu.cn

Submitted: 22nd June 2025; Accepted: 27th December 2025

Abstract

Since the proposal of the blockchain, its application scenarios have been continuously expanded. However, the anonymity feature of the blockchain has hindered market regulation, leading to numerous illegal activities such as phishing fraud, which has now become a serious type of crime. Currently, most phishing fraud detection technologies on blockchain platforms use transaction data to construct basic raw transaction graphs and then use neural network methods to mine key information. This study proposes a graph gated recurrent neural network (GGRNN) model that fully integrates temporal and spatial information, effectively utilizing time-related information in the transaction graph. It first takes an account as the center node to obtain its second-order transaction data and then constructs a dynamic transaction graph (DTG). Subsequently, the DTG is fed to the GGRNN to process the temporal features in a gated recurrent unit (GRU) framework and introduce graph convolutional network (GCN) operations to fully use the node neighbourhood topology features, obtain the embedded representation of the graph, and then perform graph classification for phishing node detection. To verify the effectiveness of the proposed model, it was applied to real-world Ethereum transaction datasets. Numerical results show that the proposed GGRNN model significantly outperforms state-of-the-art methods.

Keywords: blockchain, transaction graph, graph neural network, recurrent neural network

† These authors contributed equally.

1 Introduction

Since the concept of Bitcoin [1] was proposed by "Satoshi Nakamoto", blockchain technology has gradually gained recognition for its excellent decentralization, distributed storage, traceability, and data tamper-proof capabilities. The emergence of Ethereum has marked the maturity of blockchain technology. For example, Ethereum's smart contracts are computer programs that automate contract execution, improving the speed and efficiency of contract execution. Additionally, because Ethereum has Turing completeness, its smart contracts have greater flexibility and scalability. Moreover, the application scenarios of blockchain technology are constantly expanding, such as in electronic currency [2], financial services [3], medical insurance [4], supply chain technology [5], and distributed storage [6]. According to CoinMarket-Cap¹, more than 10,000 types of cryptocurrencies exist worldwide, with a total market value of up to \$1.14 trillion and daily global cryptocurrency trading volume of up to \$30 billion.

Because of the enormous value of blockchain, countless hackers have been attracted to attack blockchain platforms to obtain illegal profits. Moreover, because of blockchain anonymity, market regulation is hindered, leading to a significant increase in illegal activities such as price manipulation, phishing fraud, and flash loan attacks on blockchain financial platforms. Among these, phishing fraud has become the most serious type of crime on blockchain platforms, causing incalculable losses to users and platforms. According to the data compiled by SlowMist Hacked², the loss of only one type of financial currency, Ethereum, in the year 2022 amounted to more than 528 million US dollars. Therefore, financial security has become a top priority in the blockchain ecosystem, and effective phishing node identification methods must be developed to implement effective market supervision.

Although this issue is crucial, to the best of our knowledge, research in the field of blockchain phishing node identification remains relatively scarce. Existing studies mostly define node identities based on blockchain transaction account types, enabling the simultaneous identification of phish-

ing nodes alongside other identities such as miners, exchanges, and ICO wallets. Zhang et al. [7] proposed the Multi-Channel Graph Classification (MCGC) model, while Zhou et al. [8] presented the Ethident model with the Hierarchical Graph Attention Encoder (HGATE) as its core. Both models leverage graph neural networks to identify phishing nodes in transaction graphs, achieving significant performance improvements over traditional machine learning methods. Additionally, Liu et al. [9] integrated Transformer encoders with convolutional neural networks and introduced the sparrow search algorithm for hyperparameter optimization, providing a multi-dimensional fusion approach for feature extraction from graph-structured data; Ranganatha et al. [10] adopted bidirectional 3D quasi-recurrent neural networks to detect transaction fraud, whose modeling logic for temporal transaction data offers valuable insights for phishing behavior detection.

It is worth noting that security detection research in the Ethereum ecosystem also includes the important direction of smart contract vulnerability detection. For instance, Duy et al. [11] combined graph neural networks (GNNs) with language models to complete smart contract vulnerability detection; Muniswamy et al. [12] proposed a two-layer architecture combining DPoS and deep reinforcement learning, integrating CNNs to detect abnormal transaction patterns. Wang et al. [13] proposed the ContractCheck model for fine-grained vulnerability detection in smart contracts, which extracts fine-grained code snippets from abstract syntax trees and function call graphs of smart contract source code, converts them into semantic token streams, and generates numerical vector sequences. Such technologies for semantic parsing of contract code and multi-modal feature fusion provide useful references for mining behavioral features of suspicious contracts in phishing detection.

In recent years, several advanced models have further enhanced phishing detection capabilities on the Ethereum platform. Liu et al. [14] verified that transaction time density can assist in distinguishing account identities; Sui et al. [15] proposed the EPAD framework, Wang et al. [16] presented the PDTGA method, and Chen et al. [17] developed

¹<https://coinmarketcap.com/>

²<https://hacked.slowmist.io/>

the DA-HGNN model. These models address key challenges in phishing detection—including structural capture, dynamic modeling, and class imbalance—through graph contrastive learning, temporal graph attention mechanisms, and the integration of data augmentation with hybrid graph neural networks, respectively. Furthermore, the Temporal Transaction Subgraph Network (T-TSGN) proposed in [18] is used for reconstructing transaction graphs; Li et al. [19] integrated spatiotemporal dimension information to predict unknown node identities. These technologies all provide effective support for the accurate identification of phishing nodes.

Existing models still have much room for improvement. An important point is that temporal information is not fully utilized. Most existing works either do not use time information or extract time features that are too simple. In blockchain platforms, transactions among nodes have a temporal order, and the temporal information reflects the complex reality of the transaction network, without which a large amount of transaction-flow information is lost. A second point is that existing works handle temporal features at the model level relatively superficially, as most of them simply input the outputs of graph neural networks or recurrent neural networks to another model, achieving a simple concatenation of two models.

To address these two issues, this study proposes the graph gated recurrent neural network (GGRNN) model to inference the identity of transaction nodes on blockchain platforms. Initially, by splitting the transaction data in the Ethereum network into several temporal partitions, dynamic transaction graphs (DTGs) are constructed, which capture the temporal features of the entire transaction graph and improve the utilization of raw transaction data. Moreover, we designed an end-to-end GGRNN model that deeply integrates graph convolutional network (GCN) [20] with gated recurrent unit (GRU) networks [21] by defining the gated unit as a graph convolution operation. It uses the GRU as a framework to capture the temporal information of long-term dependencies [22] and uses the GCN model to compute candidate states, which can extract the topological feature of the graph data at each time step. Hence, the model comprehensively extracts temporal and spatial features and their deep

integration. The main contributions of this study are summarized as follows:

- To fully utilize temporal transaction information, a DTG is constructed by dividing the transaction data into temporal partitions.
- The GGRNN model is proposed by deeply integrating GCN and GRU networks to better capture temporal and spatial features.
- Extensive experiments on a real-world Ethereum phishing fraud dataset are conducted, and the results demonstrate that the proposed GGRNN model outperforms existing state-of-the-art (SOTA) methods.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 presents detailed designs of the GGRNN. Section 4 explains the experiments conducted and results, and Section 5 concludes the paper.

2 Related Works

2.1 Blockchain

Bitcoin emerged in 2008 and was proposed by an anonymous person or organization known as "Satoshi Nakamoto" [1]. Its appearance is the premise for the flourishing development of blockchain technology. As one of the most popular technologies today, blockchain has excellent capabilities in distributed storage, traceability, and data tamper-resistance. Blockchain decentralization means that all nodes in the entire blockchain platform jointly manage and control the entire platform, unlike traditional centralized management conducted by a specific node. In such a distributed network, all nodes must reach a consensus on transaction records and state updates. The consensus mechanism is an important component of the blockchain. In addition, because each node in the blockchain can view and verify transaction records and state updates on the network, network transparency and fairness are ensured.

The emergence of Ethereum further enhanced the performance of blockchain platforms. The most prominent features are Turing completeness and smart contracts. Ethereum smart contracts are

computer programs that can be automatically executed when required and used for managing digital assets, voting, insurance, and other scenarios. Compared with traditional contracts, smart contracts significantly improved the efficiency of contract execution. In addition, Ethereum's Turing completeness means that the platforms can handle any computable problem in computer science, making Ethereum's smart contracts more flexible and scalable.

2.2 Graph Learning

Unlike traditional data types, graph data have a non-Euclidean structure and are difficult to process using traditional methods. This leads to graph learning, which aims to learn complex relationships among nodes and the topological structure of graphs. Existing mainstream graph learning methods include graph embedding methods and graph neural networks. First, using graph embedding methods, Perozzi et al. generated paths from graph topology through random walks and then used the obtained paths as inputs to a skip-gram model to train node feature representations [23]. Grover et al. used random walks to embed graphs; however, this method could selectively bias the sequences obtained based on the parameter settings [24]. Narayanan et al. used rooted subgraphs comprising nodes and their neighbors and then used the skip-gram method to transform graph data into graph feature vector representations [25]. Tang et al. defined and calculated the first- and second-order proximities between nodes, representing the direct and indirect adjacent relationships between nodes, respectively, and maximized the proximity between nodes to generate their vector representations [26].

The history of graph neural networks can be traced back to 1997 when Sperduti et al. first applied neural networks to directed acyclic graphs and began exploring graph neural networks [27]. The main goal of graph convolutional neural networks is to apply convolution operations to graph data using a mapping function to aggregate the information contained in the nodes of the graph and their neighboring nodes to generate a new representation of the node. Initially, graph convolutional networks were primarily based on spectral models, which were mostly supported by graph signal processing methods, and different models were designed by

constructing graph signal filters. However, these methods generally suffer from the computational cost increasing with the scale of the graph and poor flexibility. After continuous improvements of graph convolutional neural networks [28, 29, 30], Kipf et al. simplified the Chebyshev polynomial fitting convolution kernel method to a lower computational complexity method [20], greatly improving the computational efficiency. Moreover, the method proposed in this study can explain graph convolutional neural networks from both the spectral and spatial domains, significantly enhancing the performance of graph convolutional neural networks in practical applications.

2.3 Dynamic Graph Neural Networks

In many practical applications, graph data are dynamic, meaning that the shape and structure of the graph and the information contained within it are not fixed but change over time. Therefore, new network structures must be designed to capture the dynamic information of these graphs.

Li et al. proposed the diffusion convolutional recurrent neural network (DCRNN), which first uses a bidirectional graph random walk method to extract spatial features and then uses a GRU framework to capture temporal dynamics [31]. Yu et al. proposed the spatio-temporal graph convolutional networks (STGCN), which consists of two spatiotemporal convolution modules (ST-Conv Block) and a fully connected output layer. Each ST-Conv Block includes a spatial graph convolution layer between two temporal gated convolution layers. The output layer outputs the final embedding expression [32]. Guo et al. proposed the attention based spatial-temporal graph convolutional network (ASTGCN) model, which uses spatiotemporal attention mechanisms to model dynamic spatiotemporal dependencies in traffic data and effectively extracts complex spatial correlations between different locations and dynamic temporal correlations between different times [33]. Song et al. proposed the spatial-temporal synchronous graph convolutional networks (STSGCN), which first converts the input features to a higher-dimensional space through the input layer and then repeats this operation to capture local spatiotemporal correlations in dynamic graphs [34]. Bai et al. proposed a method called spatial-temporal graph to sequence (STG2seq), which achieves multistep pre-

diction using both a long- and short-term encoder and models time dependencies using an attention mechanism [35]. Wu et al. proposed Graph WaveNet, which performs well in processing longer sequences by exponentially increasing the receptive field with an increase in the number of layers [36]. Pareja et al. proposed the evolving graph convolutional networks (EvolveGCN) method, which aggregates information using gated recurrent units and long short-term memory recurrent neural networks in the time dimension and then uses the obtained feature matrix as the training parameter matrix of the graph convolutional neural network to aggregate information on the topological structure of the graph, resulting in two models [37]. Li et al. encapsulated EvolveGCN into a module and added a convolution module in the time dimension to simultaneously extract the structural and temporal dynamics of the dynamic temporal graphs [38].

These models fail to fully tap into the deep potential of temporal information. In most cases, the handling of temporal features is relatively superficial—typically treated merely as part of the input data, without extracting the complex patterns embedded in the temporal dimension through more advanced mechanisms. Such models may only adopt simple linear processing or static time slicing methods, overlooking nonlinear relationships, cyclical fluctuations, or the impact of sudden events in time series. Consequently, while these models can capture historical trends reasonably well in long-term prediction tasks, they perform poorly in fine-tuning temporal dynamic changes, handling short-term disturbances, or responding to sudden events—especially when compared to models that deeply integrate temporal information, there exists a significant performance gap. To address these limitations, this study extracts the complex patterns inherent in the temporal dimension of data by designing and constructing a Dynamic Transaction Graph (DTG), thereby deeply mining nonlinear dependencies and cyclical patterns within the temporal dimension. Additionally, by processing temporal features under the Gated Recurrent Unit (GRU) framework and introducing Graph Convolutional Network (GCN) operations to fully leverage the topological features of node neighborhoods, the proposed model enhances adaptability to sudden disturbances and dynamic topological changes.

3 Methodology

3.1 Problem Description

The study aims to recognize phishing accounts in Ethereum. The raw data obtained are transactions between different accounts with amounts and timestamps. By treating account addresses as nodes in the graph and transactions as connections between nodes, the problem can be defined as a graph node classification problem.

Definition 1: Dynamic Transaction Graph (DTG)

Let $V = \{v_i \mid i = 1, 2, \dots, n\}$ be a set of account nodes in Ethereum and $E = \{e_{pq} \mid p, q \in V\}$ be the transaction set with timestamps, where e_{pq} is the edge between accounts p and q , referring to the transactions between them. Then, all transactions are evenly divided into T time slices according to the number of transactions. The DTG on V and E can be defined as

$$\mathbf{G} = \{\mathbf{G}_t\} = \{(V, E_t, X_t)\}, \quad t = 1, 2, \dots, T, \quad (1)$$

where E_t is the transaction set in the t -th time clip, and $X_t \in \mathbb{R}^{d \times n}$ is the feature matrix in the t -th time slice.

To fully use the neighborhood information, a DTG is constructed for each account by extracting its two-order neighbors and all transactions among them. Let $\mathbf{G} = \{\mathbf{G}_{add}, add \in V_{\text{Dataset}}\}$ be the DTG set of two order neighbors, where add represents the address of a node and V_{Dataset} is the set of target account nodes in the dataset and $|V_{\text{Dataset}}| = N$, and $\mathbf{Y} = \{y_{add}, add \in V_{\text{Dataset}}\}$ be the label set, with $y_{add} = 1$ indicating that the account add is a phishing account and $y_{add} = 0$ indicating that the account add is normal. This study aims to learn a mapping function $\mathbf{F} : f(\mathbf{G}) \rightarrow \mathbf{Y}$ by minimizing the loss function

$$\mathbf{L}(\mathbf{G}) = \sum_{add \in V_{\text{Dataset}}} (f(\mathbf{G}_{add}) - y_{add})^2. \quad (2)$$

The following subsections describe the construction of the DTG and method framework.

3.2 Construction of the Dynamic Transaction Graph

Our method represents each account node using a DTG constructed from its two-order transaction

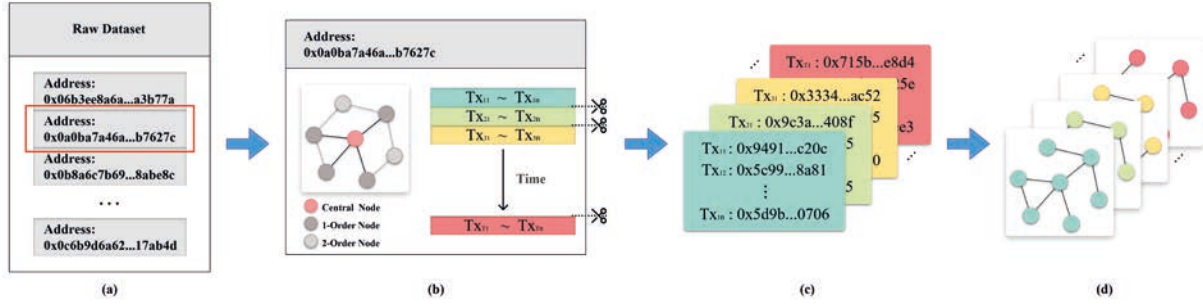


Figure 1. Construction of the dynamic transaction graph: (a) the raw transaction dataset, organized according to the account addresses; (b) the transaction data of each account address in the raw dataset is expanded to a two-order transaction dataset, arranged in chronological order; (c) the transaction data of (b) is split into temporal slices; (d) the dynamic transaction graph is constructed from the series of temporal slices in (c).

data. The DTG includes rich temporal information, which can improve the performance compared with methods that do not utilize temporal features. Figure 1 illustrates the construction process.

A piece of raw transaction data can be represented as a quarter tuple

$$q_t = (add_from, add_to, amount, timestamp),$$

whose elements are the source address, destination address, transaction amount, and timestamp, respectively. They are organized according to the address, that is, each address add corresponds to a quarter tuple dataset QTI_{add} (Figure 1(a)).

$$QTI_{add} = \{q_{t_k} \mid q_{t_k}.add_from = add \parallel q_{t_k}.add_to = add\} \quad (3)$$

To leverage the neighborhood information, the quarter tuple dataset QTI_{add} is expanded to a two-order transaction dataset (Figure 1(b)):

$$QT2_{add} = \bigcup_{k \in QTI_{add}.ADD} QTI_k \quad (4)$$

where $QTI_{add}.ADD$ is the address set affiliated with the quarter tuples in QTI_{add} :

$$QTI_{add}.ADD = \{add \mid add = q_{t_k}.add_from \parallel add = q_{t_k}.add_to, \forall q_{t_k} \in QTI_{add}\} \quad (5)$$

Then, for each two-order transaction dataset, for example $QT2_{add}$, it is split into T temporal slices Tx_{add}^t , each containing

$$|Tx_{add}^t| = \left\lfloor \frac{Tx_{add}}{T} \right\rfloor$$

transactions, where Tx_{add} is the number of transactions in $QT2_{add}$, and $\lfloor \cdot \rfloor$ denotes the rounding function (Figure 1(c)).

Subsequently, a DTG can be constructed from its split T temporal slices (Figure 1(d)). To simplify the model, multiple transactions between the same account pairs within a temporal slice are merged into a single transaction, obtaining the edge set of the t -th slice $E_{add}^{(t)}$. For each account node, a 12-dimension feature vector is abstracted for representation in the t -th slice, containing the node's in-degree, out-degree, total degree (that is, its total number of neighbors), number of incoming transactions, number of outgoing transactions, total number of transactions, total amount of incoming transactions, total amount of outgoing transactions, total transaction amount, average incoming amount, average outgoing amount, and average total transaction amount. The feature vectors of all nodes are combined to form the node feature matrix $X_{add}^{(t)}$ in the DTG at slice t . Hence, the DTG is constructed for each account address add as

$$\mathbf{G}_{add} = \{G_{add}^{(t)}\} = \left\{ \left(V, E_{add}^{(t)}, X_{add}^{(t)} \right) \right\}, \quad (6) \\ t = 1, 2, \dots, T.$$

3.3 Framework of the Graph Convolutional Gated Recurrent Network

To detect phishing nodes on a blockchain platform, this study proposes the GGRNN model. It takes the preprocessed DTG as input, represented by Eq.(6), and adopts a backbone architecture

that deeply integrates graph convolution with gated recurrent computation. Unlike a standard GRU that receives a single feature vector per time step, GGRNN processes a sequence of graph-structured states (X_t, A_t) , enabling the model to jointly capture temporal dependencies and topology-aware spatial correlations. As illustrated in Figure 2, the framework consists of three major components: (1) a GRU module enhanced with graph convolution, (2) a dual-GCN block that injects structural information into the recurrent update mechanism, and (3) an output layer that performs graph-level classification.

GRU Module

A DTG contains a series of transaction temporal slices. Therefore, the GRU network [21] is selected as the main framework of the model to extract its rich temporal features. Unlike the standard GRU network, in which the input at a single time point is a single vector, the GGRNN model uses a graph structure consisting of multiple account nodes as inputs at a single time point, thereby coupling the graph convolution operation inside the GRU. In particular, to address the graph-structure inputs, an update gate containing graph convolution is designed, leveraging the node neighborhood information in the process of updating the state.

The GRU cell state is a linear interpolation between the previous and candidate state:

$$H_t = z_t \odot H_{t-1} + (1 - z_t) \odot \tilde{H}_t, \quad (7)$$

where $H_t \in \mathbb{R}^{D \times n}$ is the GRU cell state at time t ; $\tilde{H}_t \in \mathbb{R}^{D \times n}$ is the candidate state; H_{t-1} is the previous state; n , the node number; D , the state vector dimension; $\odot$, the Hadamard product; and $z_t \in \mathbb{R}^{D \times n}$, the update gate determining how much the unit updates its state. This is computed as follows:

$$z_t = \sigma(W_z X_t + U_z H_{t-1} + b_z), \quad (8)$$

where $X_t \in \mathbb{R}^{d \times n}$ is the input at time point t (with d being the feature dimension of each node), $W_z \in \mathbb{R}^{D \times d}$, $U_z \in \mathbb{R}^{D \times D}$ are the learnable weights corresponding to X_t and H_{t-1} , and $b_z \in \mathbb{R}^{D \times n}$ is the learnable bias. Candidate state $\tilde{H}_t$ is computed by

$$\tilde{H}_t = \tanh(\hat{H}_t + b_h), \quad (9)$$

where $b_h \in \mathbb{R}^{D \times n}$ is another learnable bias, $\hat{H}_t$ corresponds to the “weighted sum” term in the standard GRU.

In the standard GRU,

$$\hat{H}_t = \hat{W}_t X_t + \hat{U}_t (r_t \odot H_{t-1}), \quad (10)$$

which is a purely linear transformation. However, because our input is a graph-structured state, this linear operation is replaced by a topology-aware graph convolution block, as detailed below.

GCN Block

To leverage neighborhood features, the input X_t and the gated previous state H_{t-1} in (10) are processed using graph convolution operators. A novel GCN block is designed to accommodate these operators; it consists of two GCN networks with the same structure but different weights (Figure 2). One network handles the input X_t , and the other handles the previous state $r_t \odot H_{t-1}$. The block is abstracted as follows:

$$\hat{H}_t = GCN_t^{(1)}(A_t, X_t) + GCN_t^{(2)}(A_t, r_t \odot H_{t-1}). \quad (11)$$

This formulation explicitly replaces the linear transformation in the GRU candidate-state computation with graph convolution, enabling each node to integrate both its temporal evolution and its structural context within the DTG at time t . Thus, $\hat{H}_t$ serves as a structure-aware counterpart of the weighted sum term in (10), and is subsequently used to compute the candidate state via (9). Here, A_t is the adjacency matrix in the t -th slice, $+$ denotes element-wise addition between the two graph-convolved representations, and $\odot$ is the Hadamard product. The reset gate r_t is computed as:

$$r_t = \sigma(W_r X_t + U_r H_{t-1} + b_r). \quad (12)$$

For completeness, the graph convolution layer inside each GCN follows:

$$H_t^{(l)} = W_t H_t^{(l-1)} \left(\tilde{D}_t^{-\frac{1}{2}} \tilde{A}_t \tilde{D}_t^{-\frac{1}{2}} \right), \quad (13)$$

where notations remain as originally defined.

Output Layer

The final output $H_T \in \mathbb{R}^{D \times n}$ of the GRU module is the feature matrix of n nodes in the DTG. Our framework models phishing account detection as a graph classification task; therefore, the output H_T of the GRU is aggregated across all nodes. Here, we

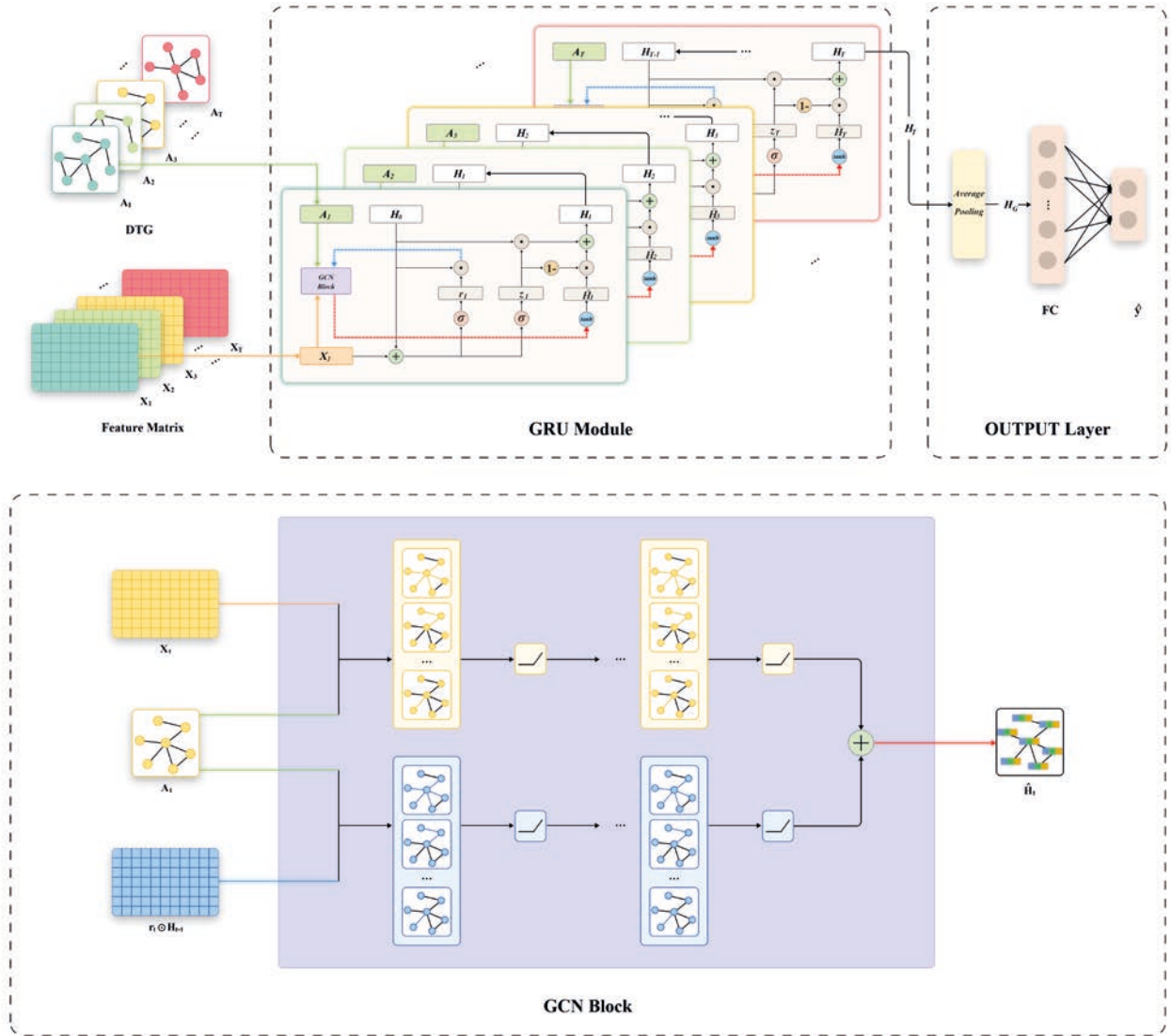


Figure 2. GGRNN framework.

Table 1. Distribution of Datasets

Dataset	N	N_p	n	n_{avg}	e_{avg}	e_{avg_raw}	N_d
D1	450	225	$0 < n < 20$	13	46	26	450
D2	450	225	$20 \leq n < 50$	32	88	68	397
D3	450	225	$50 \leq n < 200$	102	271	220	433
D4	450	225	$20 \leq n < 1000$	477	1255	1045	448

N denotes the total number of samples in the dataset; N_p , the number of positive samples in the dataset; n , the range of node numbers in the dynamic transaction graph; n_{avg} , the average number of nodes in the dynamic transaction graph; e_{avg} , the average number of edges in the dynamic transaction graph; e_{avg_raw} , the average number of edges in the static transaction graph constructed from the raw data; and N_d , the number of samples where the number of edges in the dynamic transaction graph is greater than that in the static transaction graph.

apply the average pooling operation to obtain the aggregated feature vector $H_G \in \mathbb{R}^D$, that is,

$$H_G = \text{Average_Pooling}(H_T). \quad (14)$$

A fully connected layer is utilized to map H_G to $y \in \mathbb{R}^2$ to classify the graph:

$$\hat{y} = \text{Softmax}(W_G H_G + b_G), \quad (15)$$

where W_G and b_G are parameters of the fully connected layer.

In this study, the GGRNN model uses the binary cross-entropy loss function as the training objective:

$$\mathcal{L} = -(y * \log(\hat{y})) + (1 - y) * \log(1 - \hat{y}), \quad (16)$$

where y is the identity label, and $\hat{y}$ is the predicted result.

4 Experiments and Results

4.1 Data Collection

First, we obtained second-order raw transaction data on Ethereum accounts from the open blockchain data website Xblock³. The original data contained 3360 account nodes, including 1660 positive examples (phishing nodes) and 1700 negative examples (non-phishing nodes). The data were cleaned by deleting related accounts with less than 10 transaction records or with more than 1000 two-order neighbors. Then, four datasets with different scales were constructed from the original dataset; Table. 1 lists the distributions of the datasets.

Each of the four datasets contains 450 transaction graphs, with 50% positive and 50% negative examples. Based on two core considerations—the structural characteristics of the Ethereum transaction network and the practical needs of phishing detection research: most small-value daily transaction accounts have fewer than 20 associated nodes, small and medium-sized commercial or high-frequency transaction accounts have nodes concentrated in the range of 20–200, and a small

number of large-value fund transfer or platform-type accounts have nodes up to 200–1000. Therefore, the ranges of node numbers in the DTG are (0, 20), [20, 50), [50, 200), and [200, 1000), respectively. The transaction graphs were constructed by obtaining all the first- and second-order neighboring nodes using an account node as the center of the graph. Therefore, the structure of a transaction graph is mostly star-shaped when the amount of transaction data is relatively small. However, the number of transactions between the first- and second-order nodes increases with the amount of transaction data; thus, the structure of the transaction graph may be mostly network-shaped. Table. 1 shows that the number of edges in most of the nodes of the DTG improved compared with the static transaction graph, indicating that the DTG significantly increases the amount of information.

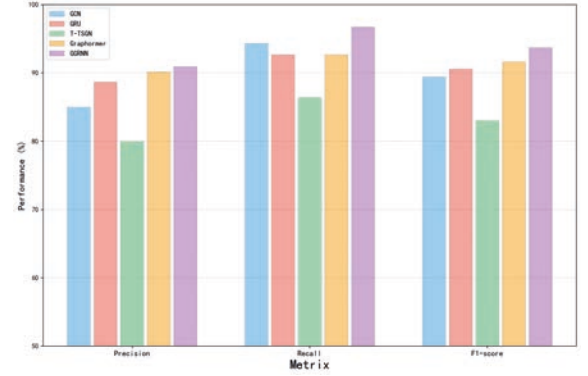


Figure 3. Comparison results of the average performances on four datasets.

4.2 Experiment Setup

To test the performance of the proposed GGRNN method, it was applied to the four datasets. For comparison, five state-of-the-art methods, DeepWalk [23], Graph2Vec [25], GCN [20], GRU [21], and T-TSGN [18], were executed on the same datasets. All experiments were conducted on a server on a Ubuntu 18.04.1 platform with an X-processor AMD Ryzen 3960X CPU and 128 GB of RAM. The GPU used for graph learning was an Nvidia RTX 3090.

For the proposed GGRNN method, the number of time slices T was set to 12; accordingly, a DTG consisting of 12 time-sliced graphs was set as the model input for each central node. The number of

³<http://xblock.pro>

Table 2. Performance Results on Four Datasets (%)

Category	Method	D1			D2			D3			D4		
		Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Embedding	DeepWalk	50.50	49.83	50.16	52.59	54.44	53.50	54.86	52.89	53.86	53.57	48.29	50.79
	Graph2Vec	73.57	65.18	69.12	71.52	60.49	65.54	61.11	60.03	60.57	71.65	85.93	78.14
Deep Learning	GCN	82.51	92.39	87.17	83.90	94.64	88.95	87.01	95.57	91.09	86.50	94.60	90.37
	GRU	90.99	90.67	90.83	88.47	92.37	90.38	83.81	93.79	88.52	91.39	93.83	92.59
	T-TSGN	92.00	88.46	90.20	85.71	78.26	81.82	84.21	88.89	86.49	57.78	100.00	73.24
	Graphormer	89.62	90.13	90.86	88.89	93.57	91.18	92.43	90.31	91.34	89.71	96.67	93.06
Ours	GGRNN	91.30	91.30	91.30	84.62	95.65	89.80	95.83	100.00	97.87	92.00	100.00	95.83

graph convolutional network layers was set to 2, hidden layer dimension to 100 and the batch size to 32. The learning rate was initialized at 0.01 and dynamically adjusted by a decreasing factor of 0.01 until it reached 10^{-6} .

The five comparison methods included two graph embedding methods, DeepWalk and Graph2Vec, and three deep learning-based methods, GCN, GRU, and the T-TSGN model. The two graph embedding methods obtain the embedding vectors of the account nodes and then use a classification algorithm to identify phishing nodes. This study selected support vector machine (SVM) as the classification algorithm. The parameter settings for the five comparison methods were as follows:

- **DeepWalk:** The step size was set to 40, the context window size to 5, the epoch to 1000, and the dimension of the output embedding to 128.
- **Graph2Vec:** The Weisfeiler-Lehman iteration number was set to 2, the down-sampling frequency to 0.0001, the minimum frequency of the graph feature occurrence to 5, the epoch to 1000, the learning rate to 0.025, and the dimension of the output embedding to 128.
- **GCN:** The number of graph convolutional network layers was set to 2, the dropout to 0.5, the batch size to 32, and the dynamically adjusted learning rate ranged from an initial value of 0.01 to 10^{-6} .
- **GRU:** The state length of the time series was set to 10, the batch size to 32, and the dynamically adjusted learning rate ranged from an initial value of 0.01 to 10^{-6} .
- **T-TSGN:** T-TSGN is a deep learning-based method that uses diffpool pooling and temporal information [39]. In the experiments of [18], T-TSGN outperformed all the methods compared. We retained the parameter settings from [18] and

constructed the graph topology from the original data.

- **Graphormer:** The hidden dimension is set to 100, the number of attention heads to 3, and the batch size to 32, and the dynamically adjusted learning rate ranged from an initial value of 0.01 to 10^{-6} .

All results were based on ten-fold cross-validated statistics. The precision, recall, and F1-score were selected as evaluation metrics for the comparison and were calculated as follows:

$$P = \frac{TP}{TP + FP}, \quad (17)$$

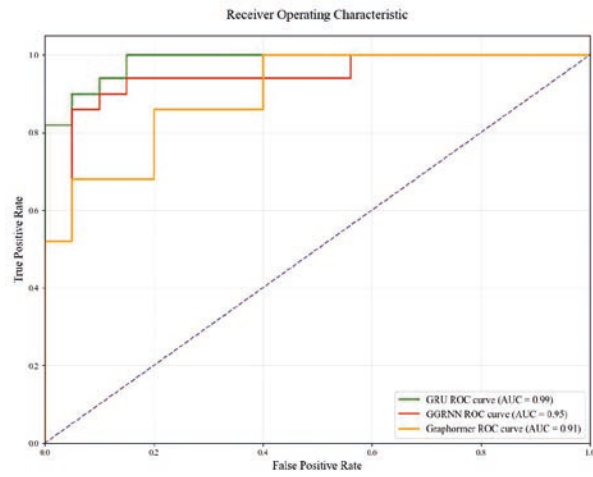
$$R = \frac{TP}{TP + FN}, \quad (18)$$

$$F_1 = \frac{2PR}{P + R}, \quad (19)$$

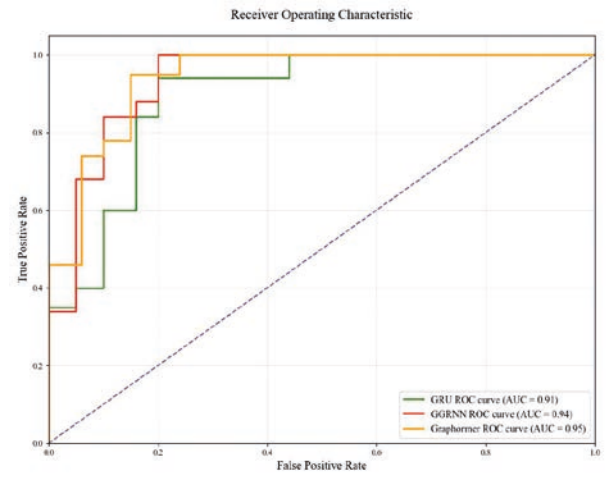
where P refers to precision, R to recall, TP stands for the true positives (the number of samples that the model correctly predicts as phishing nodes with their true identities), FP stands for the false positives (the number of samples that the model predicts as phishing nodes incorrectly), FN stands for the false negatives (the number of samples that the model fails to predict as phishing nodes when their true identities are phishing nodes), and the F_1 -score is the harmonic mean of the precision and recall. The F_1 -score can provide a more comprehensive determination of the classification model's performance.

4.3 Results

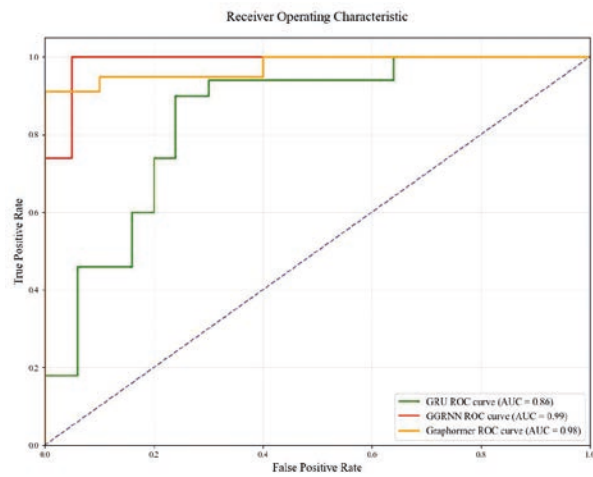
Table. 2 compares the performance results on the four datasets. Evidently, the two embedding-based methods are significantly less effective than



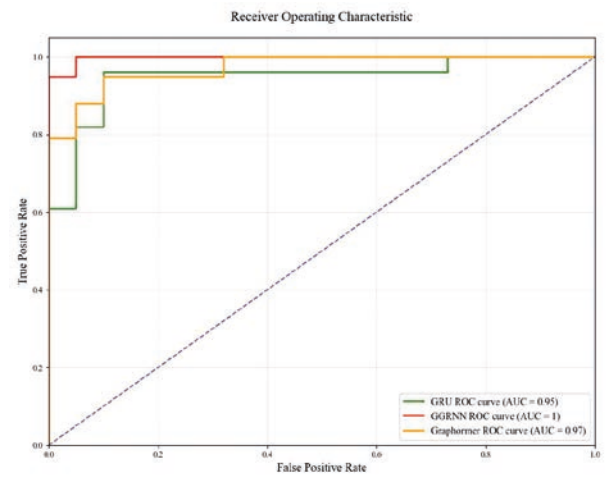
(a) D1



(b) D2



(c) D3



(d) D4

Figure 4. Comparative ROC AUC analysis across four datasets: (a) D1, (b) D2, (c) D3, (d) D4.

other deep learning-based methods. Taking the F1-score on Dataset1 as an example, Graph2Vec obtains 69.12%, which is 25.23% lower than that of the GCN, which is the worst performer among the deep learning-based methods. Another graph embedding method, DeepWalk, obtained a lower F1-score of 50.16%. The results on the other datasets and performances in terms of the precision and recall metrics are similar. Among the 12 core metrics across the four datasets, the GGRNN secures the top performance in 8 of them (Recall + F1-score on D1, Precision + F1-score on D2, Precision + F1-score on D3, and Precision + F1-score on D4). Notably, it does not rank the lowest in any metric, demonstrating exceptional stability and overall performance. Furthermore, the GGRNN exhibits remarkable balance: its precision consistently ranges between 84% and 96%, while its recall falls within 91% to 100%. This eliminates significant trade-offs, avoiding the extreme scenario of "high recall but low precision" observed in T-TSGN.

Taking the most critical F1-score metric as an example, the GGRNN achieved 91.30%, 89.80%, 97.87%, and 95.83% on the four datasets, which are the best performances among the methods, and improved by 0.44%, -1.38%, 6.53%, and 2.77%, respectively, compared with the best performing comparison method. The GGRNN achieved impressive results in terms of the precision and recall metrics, surpassing the best-performing comparison method by an average of 0.067% and 1.086%, respectively. Figure 3 shows the comparison results of the average performance on the four datasets. In terms of the average performance of the three metrics (precision, recall, and F1-score), the GGRNN ranks first, outperforming the best-performing comparison method by 0.78%, 2.44%, and 2.09% respectively.

As shown in Figure 4, GGRNN's ROC curves consistently lie above those of baselines on all four datasets, yielding ROC AUC scores of 0.95 (D1), 0.94 (D2), 0.95 (D3), and 1.00 (D4). These results confirm GGRNN's superior ability to distinguish phishing from legitimate accounts across imbalanced settings.

4.4 Ablation Analysis

4.4.1 Number of Time Slices

To evaluate the model performance under different time slice numbers (T), we conducted ablation experiments on D1–D4 with $T \in \{8, 10, 12, 14\}$, using Precision (P), Recall (R), and F1-score in Table 3.

Table 3. Model performance with varying numbers of time slices

Time slices	Metric	Datasets			
		D1	D2	D3	D4
8	Precision	95.65	85.19	91.30	81.48
	Recall	95.65	100.00	91.30	95.65
	F1-score	95.65	92.00	91.30	88.00
10	Precision	91.80	91.75	91.24	92.80
	Recall	92.43	92.02	91.19	94.25
	F1-score	92.11	91.88	91.21	93.52
12	Precision	91.30	84.62	95.83	92.00
	Recall	91.30	95.65	100.00	100.00
	F1-score	91.30	89.80	97.87	95.83
14	Precision	95.65	90.48	95.83	90.91
	Recall	95.65	82.61	100.00	86.96
	F1-score	95.65	86.36	97.87	88.89

T significantly impacts F1-scores: D1 peaks at $T = 8/14$ (95.65%), D2 at $T = 10$ (91.88)—noting $T = 8$ for D2 has misleading F1 ($R = 100\%$ but $P = 85.19\%$)—and D3/D4 at $T = 12$ (97.87%/95.83%). Notably, $T = 12$ maintains stable high F1 (89.80%–97.87%) across all datasets, outperforming $T = 8/12/14$ (strong only on individual datasets). For critical P-R balance in phishing detection, $T = 12$ achieves the smallest coefficient of variation (P: 84.62%–95.83%, R: 91.30%–100.00%), while $T = 8$, $T = 12$, $T = 14$ suffer imbalances from coarse slicing, or disrupted continuity. Cross-dataset stability (essential for blockchain) is confirmed by average F1-score: $T = 12$ has the highest average score (93.70%), outperforming $T = 8$ (91.74%), $T = 10$ (92.18%), $T = 14$ (92.19%).

In conclusion, $T = 12$ stands out as the optimal hyperparameter. It delivers robust performance across multiple datasets while excelling in practical applicability—characterized by a balanced precision-recall trade-off—and generalization abil-

ity, as evidenced by its stable cross-dataset performance.

4.4.2 Performance Analysis of Hidden Layer Dimensions

Table 4 evaluates the model's precision, recall, and F1-score across four datasets (D1-D4) with hidden layer dimensions 50, 100, and 150. Key findings are as follows:

Table 4. Model Performance Under Different Hidden Layer Dimension Settings

Hidden Dim.	Metric	D1	D2	D3	D4
50	Precision	91.67	84.00	79.31	92.00
	Recall	95.65	91.30	100.00	100.00
	F1-score	93.62	87.50	88.46	95.83
100	Precision	91.30	84.62	95.83	92.00
	Recall	91.30	95.65	100.00	100.00
	F1-score	91.30	89.80	97.87	95.83
150	Precision	91.30	85.71	82.14	92.00
	Recall	91.30	78.26	100.00	100.00
	F1-score	91.30	81.81	90.19	95.83

- **50 dimensions:** Exhibits significant performance fluctuation (F1-score std = 3.56), with D2's recall dropping to 91.30% (F1-score = 87.50%) and D3's F1-score decreasing to 88.46%. Insufficient model capacity restricts the extraction of complex features, resulting in sub-optimal fitting on D2 and inadequate generalization ability on D3.
- **100 dimensions:** Achieves the most balanced and stable performance. Across all datasets, the proposed method achieves the highest mean values among the three metrics—F1-score, precision, and recall—with respective scores of 90.94%, 96.74%, and 93.70%. Furthermore, precision and recall exhibit a high degree of alignment.
- **150 dimensions:** Performance degrades with reduced precision and recall (e.g., D3 dataset precision = 52.14%, D2 dataset recall = 78.26%), along with increased F1-score fluctuation (std = 5.06). The overall performance of the model across the four datasets is inferior to that when using 50 dimensions.

In summary, 100 dimensions is the optimal hidden layer configuration, balancing performance, stability, and generalization.

4.5 Parameter Count and Speed

Table 5. Comparison of Model Complexity and Inference Efficiency

Model	Parameter	Throughput	Inference Time
GRU	34k+	4534.27	0.0002
Graphormer	172k+	376.93	0.0025
GGRNN	47k+	1402.63	0.0007

To comprehensively evaluate model complexity and inference efficiency, three core metrics are compared: Parameter count, Throughput, and Inference Time. Parameter count reflects structural complexity (total trainable parameters, unit: thousands), while Throughput (samples/s) and Inference Time (s/sample) jointly measure computational efficiency and real-time performance.

As shown in Table 5, the three models differ significantly in parameter scale and inference performance. GRU, with a concise structure, achieves the fastest inference time (0.0002 s/sample) and highest throughput (4534.27 samples/s). In contrast, Graphormer (172k+ parameters) exhibits the slowest speed (0.0025 s/sample) and lowest throughput (376.93 samples/s), incurring higher computational overhead.

GGRNN strikes an excellent trade-off: with a moderate parameter count (47k+), it maintains competitive throughput (1402.63 samples/s) and reasonable inference time (0.0007 s/sample). Notably, despite GRU's fewer parameters and faster speed, previous experiments show GGRNN outperforms it significantly in core tasks (e.g., phishing node identification accuracy). This "moderate complexity-efficient inference-superior performance" balance highlights the practical value and application potential of the proposed GGRNN.

4.6 Visualization

The GGRNN model exhibits a strong feature extraction capability, which first extracts the embedding feature vector H_T through the GRU framework (Figure 2) and then classifies the accounts using the softmax function in the output layer. To

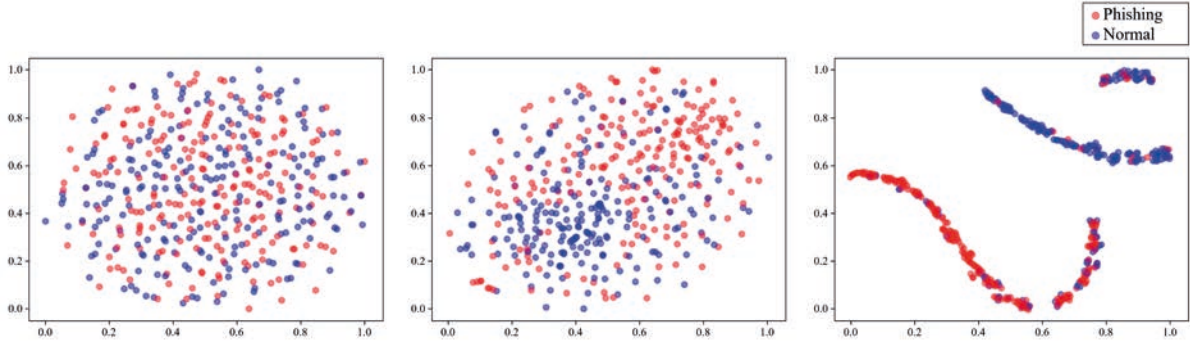


Figure 5. Visualization of nodes with different labels in a two-dimensional space. (a) DeepWalk. (b) Graph2Vec. (c) GGRNN.



Figure 6. Visualization of phishing and normal transaction pattern graphs. (a) Normal transaction pattern graphs. (b) Phishing transaction pattern graphs.

examine its feature extraction ability, it was visually compared with two graph embedding methods. The 128-dimensional embedding representation obtained by each method was reduced to two dimensions by applying T-distributed Stochastic Neighbor Embedding (t-SNE) [40] and then visualized in a two-dimensional space. Figure 5 shows the visualization results of DeepWalk, Graph2Vec, and GGRNN, where the blue points represent non-phishing account nodes and red points represent phishing account nodes. A good embedding representation should maintain the points of different colors as distant as possible and the points of the same color as close together as possible. It can be observed that the Graph2Vec method clusters the nodes better than DeepWalk to some extent, whereas GGRNN performs significantly better, separating the vast majority of the nodes by category with large separation distances between categories.

In addition, the transaction topologies of phishing and non-phishing nodes were visualized to further investigate the transaction behavior patterns of phishing nodes. Figure 6 presents the visualization results; Figure 6(a) shows two examples of trans-

action patterns for non-phishing nodes, and Figure 6(b) shows two examples of transaction patterns for phishing nodes. The results show that phishing accounts have relatively few first-order transaction nodes compared with regular accounts. However, the second-order transaction nodes associated with phishing nodes typically have a greater extent, meaning that the transaction volume of these nodes is much higher than that of other nodes in the same graph. Based on work by Zhou et al. [8], we surmise that they are typically exchanges used for asset management and currency exchange. In addition, the graph reveals that in the trading behavior patterns of phishing nodes, nodes with a high degree tend to form larger spherical structures around nodes with similar trading behaviors, as shown by the first phishing node trading pattern in Figure 6(b).

4.7 Discussion

The DeepWalk and Graph2Vec embedding methods performed poorly in this task. A possible reason is that these two methods, which aim to learn

an embedded representation of the graph, are more focused on extracting the structural information of the graph and unable to effectively mine the transaction information contained in the graph.

The GCN is a spatial graph neural network approach; thus, as the size of the transaction graph increases in the dataset, it can learn more information and therefore the performance tends to improve, as shown in Table. 2, where the performance changes from Dataset1 to Dataset3. However, the performance of the GCN on Dataset4 slightly decreases, which may be because the size distribution of the transaction graphs in Dataset4 is more discrete, which affects the model performance.

The GRU is a temporal deep learning model. The results in Table 2 show that the F1-scores of the GRU are better than those of the other methods compared, except on Dataset3. Therefore, considering the temporal order in which transactions occur effectively improves the performance of the node identification task to some extent.

T-TSGN extracts temporal features by retaining directed edges in chronological order; however, this is a manner of transforming temporal information into static features retained in the graph topology. Neighborhood information is then extracted using the Diffpool operation. The T-TSGN method utilizes both temporal and neighborhood features; however, it performs poorly on some datasets, primarily because the temporal information is modelled as a static feature, which fails to be tightly coupled with spatial information. Conversely, the GGRNN achieves a superior performance by deeply integrating temporal and spatial information at the model implementation level.

Evidently, the proposed GGRNN model performs best in identifying phishing nodes, indicating that it significantly improves the utilization of temporal information through the DTG for temporal feature representation and the GRU framework for temporal feature processing, and by fully using the topological information of the neighborhood node features through the introduction of the graph convolution operation.

5 Conclusion

For increasingly well-developed blockchains, security issues have become an unignorable major problem. To address the phishing problem on the Ethereum platform that occurs every year, an end-to-end GGRNN model for identifying phishing nodes in Ethereum was developed. Compared with previous works, this study models temporal dimension information as a DTG on timestamps, thereby significantly improving the use of temporal features and utilization of raw transaction records. In addition, we designed a GRU network framework by introducing a graph convolution operation. The GRU framework can handle temporal features effectively, and the graph convolution operation can fully use the node neighborhood topology features. To test the performance of the proposed GGRNN model, it was compared with existing SOTA methods on real datasets. The numerical results show that it achieves significantly better results.

Acknowledgment

The work is supported in part by NSFC (62272192), the Science-Technology Development Plan Project of Jilin Province (20230201073GX).

References

- [1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, 2008.
- [2] V. Buterin, "Ethereum: A next-generation cryptocurrency and decentralized application platform," *Bitcoin Magazine*, 2014.
- [3] Y. Liu, Z. Ai, S. Sun, S. Zhang, Z. Liu, and H. Yu, FedCoin: A Peer-to-Peer Payment System for Federated Learning, in *Federated Learning - Privacy and Incentive (Lecture Notes in Computer Science, Vol. 12500)*, Springer, pp. 125–138, 2020. DOI: https://doi.org/10.1007/978-3-030-63076-8_9.
- [4] H. Sami Ullah and S. Aslam, Blockchain in Healthcare and Medicine: A Contemporary Research of Applications, Challenges, and Future Perspectives, in *CoRR*, Vol. abs/2004.06795, 2020. arXiv: <https://arxiv.org/abs/2004.06795>, eprint: 2004.06795.
- [5] T. Alam and M. Benaida, Blockchain, Fog and IoT Integrated Framework: Review, Architecture and

- Evaluation, in CoRR, Vol. abs/2006.03596, 2020. arXiv: <https://arxiv.org/abs/2006.03596>, eprint: 2006.03596.
- [6] D. Perard, L. Gicquel, and J. Lacan, BlockHouse: Blockchain-Based Distributed Storehouse System, in 9th Latin-American Symposium on Dependable Computing (LADC 2019), Natal, Brazil, November 19-21, 2019, IEEE, pp. 1–4, 2019. DOI: <https://doi.org/10.1109/LADC48089.2019.8995675>.
- [7] D. Zhang, J. Chen, and X. Lu, Blockchain Phishing Scam Detection via Multi-channel Graph Classification, in Blockchain and Trustworthy Systems - Third International Conference, BlockSys 2021 (Communications in Computer and Information Science, Vol. 1490), edited by Hong-Ning Dai, Xuanzhe Liu, Daniel Xiapu Luo, Jiang Xiao, and Xiangping Chen, Guangzhou, China, August 5-6, 2021, Revised Selected Papers, Springer, pp. 241–256, 2021. DOI: https://doi.org/10.1007/978-981-16-7993-3_19.
- [8] J. Zhou, C. Hu, J. Chi, J. Wu, M. Shen, and Q. Xuan, Behavior-Aware Account De-Anonymization on Ethereum Interaction Graph, in IEEE Trans. Inf. Forensics Secur., Vol. 17, pp. 3433–3448, 2022. DOI: <https://doi.org/10.1109/TIFS.2022.3208471>.
- [9] Y. Liu, C. Wang, and Y. Ma, DL4SC: a novel deep learning-based vulnerability detection framework for smart contracts, Automated Software Engineering, Vol. 31, p. 24, 2024. DOI: <https://doi.org/10.1007/s10515-024-00418-z>.
- [10] H. R. Ranganatha and A. Syed Mustafa, Enhancing fraud detection efficiency in mobile transactions through the integration of bidirectional 3d Quasi-Recurrent Neural network and blockchain technologies, Expert Systems with Applications, Vol. 260, p. 125179, 2025. DOI: <https://doi.org/10.1016/j.eswa.2024.125179>.
- [11] P. T. Duy, N. H. Khoa, N. H. Quyen, L. C. Trinh, V. T. Kien, T. M. Hoang, and V.-H. Pham, Vulnsense: efficient vulnerability detection in ethereum smart contracts by multimodal learning with graph neural network and language model, International Journal of Information Security, Vol. 24, No. 1, p. 48, 2025. DOI: <https://doi.org/10.1007/s10207-024-00965-2>.
- [12] A. Muniswamy and R. Rathi, Trust-Based Consensus and ABAC for Blockchain Using Deep Learning to Secure Internet of Things, Applied Artificial Intelligence, Vol. 39, No. 1, 2025. DOI: <https://doi.org/10.1080/08839514.2025.2459461>.
- [13] X. Wang, S. Tian, and W. Cui, ContractCheck: Checking Ethereum Smart Contracts in Fine-Grained Level, IEEE Transactions on Software Engineering, Vol. 50, No. 7, pp. 1789–1806, 2024. DOI: <https://doi.org/10.1109/TSE.2024.3400294>.
- [14] X. Liu, Z. Tang, P. Li, S. Guo, X. Fan, and J. Zhang, A Graph Learning Based Approach for Identity Inference in DApp Platform Blockchain, in IEEE Trans. Emerg. Top. Comput., Vol. 10, No. 1, pp. 438–449, 2022. DOI: <https://doi.org/10.1109/TETC.2020.3027309>.
- [15] H. Sui, J. Zhang, B. Chen, D. Wu, X. Sun, and S. Palaiahnakote, EPAD: Ethereum phishing scam detection via graph contrastive learning, in Expert Systems with Applications, Vol. 288, Art. No. 128227, 2025. DOI: <https://doi.org/10.1016/j.eswa.2025.128227>.
- [16] L. Wang, M. Xu, and H. Cheng, Phishing scams detection via temporal graph attention network in Ethereum, in Information Processing & Management, Vol. 60, No. 4, Art. No. 103412, 2023. DOI: <https://doi.org/10.1016/j.ipm.2023.103412>.
- [17] Z. Chen, S.-Z. Liu, J. Huang, Y.-H. Xiu, H. Zhang, and H.-X. Long, Ethereum Phishing Scam Detection Based on Data Augmentation Method and Hybrid Graph Neural Network Model, in Sensors, Vol. 24, No. 12, Art. No. 4022, 2024. DOI: <https://doi.org/10.3390/s24124022>.
- [18] J. Wang, P. Chen, S. Yu, and Q. Xuan, “TSGN: Transaction subgraph networks for identifying Ethereum phishing accounts,” in BlockSys 2021, Springer, Singapore, pp. 187–200, 2021. DOI: https://doi.org/10.1007/978-981-16-7993-3_15.
- [19] S. Li, G. Gou, C. Liu, C. Hou, Z. Li, and G. Xiong, TTAGN: Temporal Transaction Aggregation Graph Network for Ethereum Phishing Scams Detection, in Proceedings of the ACM Web Conference 2022 (WWW’22), ACM, pp. 661–669, 2022. DOI: <https://doi.org/10.1145/3485447.3512226>.
- [20] T. N. Kipf and M. Welling, Semi-Supervised Classification with Graph Convolutional Networks, in 5th International Conference on Learning Representations (ICLR 2017), Toulon, France, April 24-26, 2017, OpenReview.net, 2017. URL: <https://openreview.net/forum?id=SJU4ayYgl>.
- [21] K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), edited by A. Moschitti, B. Pang, and W. Daelemans, Doha, Qatar, October

- 25-29, 2014, ACL, pp. 1724–1734, 2014. DOI: <https://doi.org/10.3115/v1/d14-1179>.
- [22] J. Chung, Ç. Gülçehre, K. H. Cho, and Y. Bengio, Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, in CoRR, Vol. abs/1412.3555, 2014. arXiv: <http://arxiv.org/abs/1412.3555>.
- [23] B. Perozzi, R. Al-Rfou, and S. Skiena, DeepWalk: Online Learning of Social Representations, in Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14), ACM, New York, NY, USA, August 24-27, 2014, pp. 701–710, 2014. DOI: <https://doi.org/10.1145/2623330.2623732>.
- [24] A. Grover and J. Leskovec, node2vec: Scalable Feature Learning for Networks, in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16), Assoc Comp Machinery, San Francisco, CA, USA, August 13-17, 2016, pp. 855–864, 2016. DOI: <https://doi.org/10.1145/2939672.2939754>.
- [25] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, graph2vec: Learning Distributed Representations of Graphs, in arXiv preprint, arXiv:1707.05005 [cs.AI], 2017. DOI: <https://doi.org/10.48550/arXiv.1707.05005>.
- [26] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, LINE: Large-scale Information Network Embedding, in Proceedings of the 24th International Conference on World Wide Web (WWW 2015), ACM, Florence, Italy, May 18-22, 2015, pp. 1067–1077, 2015. DOI: <https://doi.org/10.1145/2736277.2741093>.
- [27] A. Sperduti and A. Starita, Supervised neural networks for the classification of structures, in IEEE Trans. Neural Networks, Vol. 8, No. 3, pp. 714–735, 1997. DOI: <https://doi.org/10.1109/72.572108>.
- [28] M. Gori, G. Monfardini, and F. Scarselli, A new model for learning in graph domains, in IEEE International Joint Conference on Neural Networks (IJCNN 2005), Montreal, QC, Canada, July 31 - August 4, 2005, IEEE, pp. 729–734, 2005. DOI: <https://doi.org/10.1109/IJCNN.2005.1555942>.
- [29] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, Spectral networks and locally connected networks on graphs, in arXiv preprint, arXiv:1312.6203, 2013. URL: <https://arxiv.org/pdf/1312.6203>.
- [30] M. Defferrard, X. Bresson, and P. Vandergheynst, Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering, in Advances in Neural Information Processing Systems 29 (NeurIPS 2016), edited by D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, Barcelona, Spain, December 5-10, 2016, pp. 3837–3845, 2016. URL: <https://proceedings.neurips.cc/paper/2016/hash/04df4d434d481c5bb723be1b6df1ee65-Abstract.html>.
- [31] Y. Li, R. Yu, C. Shahabi, and Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, in arXiv preprint, arXiv:1707.01926, 2017. DOI: <https://doi.org/10.48550/arXiv.1707.01926>.
- [32] B. Yu, H. Yin, and Z. Zhu, Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting, in Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI 2018), edited by J. Lang, Stockholm, Sweden, July 13-19, 2018, pp. 3634–3640, 2018.
- [33] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting, in Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI 2019), AAAI, Honolulu, HI, USA, January 27 - February 1, 2019, pp. 922–929, 2019.
- [34] C. Song, Y. Lin, S. Guo, and H. Wan, Spatial-Temporal Synchronous Graph Convolutional Networks: A New Framework for Spatial-Temporal Network Data Forecasting, in Thirty-Fourth AAAI Conference on Artificial Intelligence, the Thirty-Second Innovative Applications of Artificial Intelligence Conference and the Tenth AAAI Symposium on Educational Advances in Artificial Intelligence (AAAI 2020), New York, NY, USA, February 7 - 12, 2020, Association for the Advancement of Artificial Intelligence (AAAI), pp. 914–921, 2020.
- [35] L. Bai, L. Yao, S. S. Kanhere, X. Wang, and Q. Z. Sheng, STG2Seq: Spatial-Temporal Graph to Sequence Model for Multi-step Passenger Demand Forecasting, in Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI 2019), Macao, Peoples R China, August 10 - 16, 2019, International Joint Conference on Artificial Intelligence (IJCAI), pp. 1981–1987, 2019.
- [36] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, Graph WaveNet for Deep Spatial-Temporal Graph Modeling, in Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI 2019), Macao, Peoples R China, August 10 - 16, 2019, International Joint Conference

on Artificial Intelligence (IJCAI), pp. 1907–1913, 2019.

- [37] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. B. Schardl, and C. E. Leiserson, EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs, in Thirty-Fourth AAAI Conference on Artificial Intelligence, the Thirty-Second Innovative Applications of Artificial Intelligence Conference and the Tenth AAAI Symposium on Educational Advances in Artificial Intelligence (AAAI 2020), New York, NY, USA, February 7 - 12, 2020, Association for the Advancement of Artificial Intelligence (AAAI), pp. 5363–5370, 2020.
- [38] J. Li, Y. Liu, and L. Zou, DynGCN: A Dynamic Graph Convolutional Network Based on Spatial-Temporal Modeling, in Web Information Systems Engineering, WISE 2020, Pt I (Lecture Notes in Computer Science, Vol. 12342), Virtual Conference, October 20 - 24, 2020, Springer, pp. 83–95, 2020. DOI: https://doi.org/10.1007/978-3-030-62005-9_7.
- [39] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, Hierarchical Graph Representation Learning with Differentiable Pooling, in Advances in Neural Information Processing Systems 31 (NIPS 2018) (Advances in Neural Information Processing Systems, Vol. 31), Montreal, QC, Canada, December 2 - 8, 2018, Curran Associates, Inc., 2018.
- [40] L. van der Maaten and G. Hinton, Visualizing Data using t-SNE, Journal of Machine Learning Research, Vol. 9, pp. 2579–2605, November 2008.



Zhaohuang Chen received the B.Sc. degree in Computer Science and Technology from Jilin University, China, in 2022. He is currently pursuing the M.Sc. degree in Computer Science and Technology at Jilin University, China, and has been awarded the Graduate Academic Scholarship. His research interests include pattern recognition and

machine learning, with a focus on image generation, graph neural networks, and related areas.

<https://orcid.org/0009-0000-7759-2751>



Zhongqi Fu received the B.Sc. degree in Computer Science and Technology from Jilin University, China, in 2020, and the M.Sc. degree in Computer Science and Technology from Jilin University, China, in 2023. His research interests include pattern recognition and machine learning, with a focus on graph neural networks, blockchain, and related areas.

<https://orcid.org/0009-0001-4050-6458>



Tao Liang received the bachelor's degree from Harbin Institute of Technology, Harbin, China, in 2022. He is currently working toward the master's degree in the College of Software, Jilin University, Changchun, China. His research interests lie in blockchain, federated learning and ensemble learning.

<https://orcid.org/0009-0009-9379-0155>



Haidong Ma received the bachelor's degree from Harbin Institute of Technology, Harbin, China, in 2022. He is currently working toward the master's degree in the College of Software, Jilin University, Changchun, China. His research interests lie in blockchain, federated learning and ensemble learning.

<https://orcid.org/0009-0001-9145-7676>



Yanfeng Sun received his Ph.D. degree from Jilin University, Jilin China in 2003. From 2004 to 2008, he was a postdoctoral fellow with Jilin University. He is now a full Associate Professor at College of Computer Science and Technology, Jilin University, Jilin China. Dr Sun has published 30+ peer-reviewed papers, including those

in highly regarded journals such as Neural Computing & Applications, Journal of universal computer science, Progress in Natural Science and Acta Physica Sinica, etc. His current research interests include complex networks, deep learning, blockchain, etc.

<https://orcid.org/0009-0009-9832-5084>



Xiaohu Shi received his M.Sc. degree in Fluid Mechanics, and his Ph.D. degree in Computer Application Technology from Jilin University, China, in 2002 and 2006, respectively. He is currently a professor in the College of Computer Science and Technology, Jilin University, China. He has published more than 80 journal and conference

papers. His current research interests include machine learning and blockchain.

<https://orcid.org/0000-0002-5115-8137>