SAMamba: Structure-Aware Mamba for Ethereum Fraud Detection

Teng Huang, Jiahui Huang, Changyu Dong, Sisi Duan, *Member, IEEE*, and Yan Pang, *Member, IEEE*

***Abstract*—The pseudonymity nature of Ethereum provides a protective umbrella for criminal activities, allowing criminals to develop a series of black industries such as phishing scams in unregulated areas. In order to exploit the relational inductive bias to discover the real identity of anonymous accounts, graph neural networks (GNNs) have been widely used in Ethereum fraud detection tasks as an effective and powerful framework. However, the expressive power of GNN's 1-hop message passing mechanism is bounded by the Weisfeiler-Leman (1-WL) test, degrading the fraud detection performance on the Ethereum network. This paper proposes a structure-aware Mamba framework, named SAMamba. Specifically, SAMamba uses a subgraph encoding strategy to capture complex structural patterns and introduces Mamba's exceptional sequence modeling capabilities to route global information. In order to filter task-relevant information from dense information, the attention mechanism and the selection mechanism are introduced from local and global perspectives, respectively. These tailor-made designs enable SAMamba to distinguish subtle differences in structural patterns and selectively aggregate task-oriented information, thereby demonstrating exceptional performance in fraud detection tasks. Extensive experiments on real-world Ethereum data demonstrate that SAMamba outperforms state-of-the-art methods. The codes are publicly available on Github: https://github.com/deepang-ai/SAMamba**

***Index Terms*—Blockchain, mamba, ethereum, graph neural network, fraud detection.**

I. Introduction

After more than a decade of iterative development, blockchain-based decentralized finance (DeFi) has achieved significant growth and formed a mature application system. As a blockchain platform at the forefront of the market, Ethereum supports peer-to-peer transactions and program execution in the form of smart contracts, allowing developers to build decentralized applications (DApps) based on smart contracts and support diversified trading businesses [1], [2]. The comprehensive and complete service system has enabled Ethereum to attract a large number of users to participate and accumulate a large amount of investment. According to data provided by defillama,[1] as of August 2024, Ethereum's total locked value (TVL) has reached US$50.5 billion, accounting for the highest 58.4% TVL in the DeFi sector.

Unfortunately, the open and collaborative nature of Ethereum has also created an environment where illegal activities can flourish. According to the 2024 Cryptocurrency Crime Report released by Chainalysis,[2] phishing scams have become the most rampant criminal activity in the cryptocurrency sector, with related wallets generating at least $4.6 billion in fraud revenue in 2023. Such scammers abuse the anonymity of Ethereum to implement more complex and diverse strategies, making it increasingly difficult to identify addresses associated with crypto scams. On Ethereum and many other blockchain platforms, identities are usually represented by long strings derived from a user's private key. This pseudonymous address itself does not directly reveal the user's real identity, allowing users to transact and interact privately and securely on the Ethereum network. As a double-edged sword, the pseudonymity of Ethereum accounts also provides a safeguard for illegal transactions, allowing the culprits to evade the tracking of regulators.

As a result, Ethereum fraud detection has attracted widespread attention, forming two mainstream technical routes and continuously iterating. One mainstream technique involves machine learning (ML) methods [4], [5], [6], [7], [8], [9] based on hand-crafted features. This is a complex and challenging task that involves using supervised learning to analyze the latent distribution of data in the Ethereum network to infer the real identity of an account. The supervised learning paradigm of ML relies on learning engineered features and identifying real identities by using the learned domain-specific knowledge. However, simply classifying based on engineered features ignores the interactions between accounts. To address this issue, the popular architecture of graph machine learning, graph neural networks (GNNs), was applied to the fraud detection task [10], [11], [12], [13], [14], [15]. GNNs follow a standard local message passing paradigm, iteratively aggregating and updating the state of a node based on mes-

Received 16 December 2024; revised 29 May 2025; accepted 7 July 2025. Date of publication 14 July 2025; date of current version 21 July 2025. This work was supported in part by the National Key Research and Development Program of China under Grant 2023YFB2704300, in part by Guangdong Natural Science Foundation under Grant 2025A1515010276, and in part by the Science and Technology Projects in Guangzhou 2024 Guangzhou School (Institute) Enterprise Joint Funding Project under Grant 2024A03J0166. The associate editor coordinating the review of this article and approving it for publication was Dr. Paolo Gasti. *(Corresponding authors: Jiahui Huang; Yan Pang.)*

Teng Huang, Jiahui Huang, and Changyu Dong are with the School of Artificial Intelligence, Guangzhou University, Guangzhou 511370, China (e-mail: huangteng1220@gzhu.edu.cn; jiahuihuang@gzhu.edu.cn; Changyu.dong@gzhu.edu.cn).

Sisi Duan is with the Institute for Advanced Study, Tsinghua University, Beijing 100190, China (e-mail: duansisi@tsinghua.edu.cn).

Yan Pang is with Guangdong Provincial Key Laboratory of Computer Vision and Virtual Reality Technology, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China (e-mail: yanpang@siat.ac.cn).

Digital Object Identifier 10.1109/TIFS.2025.3589015

[1] https://defillama.com/

[2] https://demo.chainalysis.com/

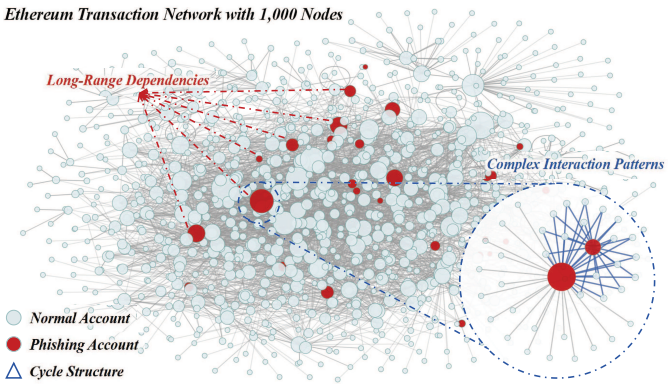


Fig. 1. The topology of the real-world Ethereum account interaction network. Data originate from the EPTransNet [3], where a local network comprising 1,000 nodes extracted via random walk sampling is presented. A force-directed layout algorithm is employed to guarantee faithful representation of the network's topological properties.

sages from its neighbors. Learning topological features allows GNNs to integrate richer information than conventional ML methods, and has shown remarkable capabilities in processing Ethereum account interaction graphs.

However, the application of state-of-the-art GNN in Ethereum account interaction scenarios still poses the following challenges: **1) Encoding of complex interaction patterns.** As depicted in Fig. 1, phishing accounts in real-world scenarios manifest intricate interaction patterns, where two neighboring nodes of the target node engage in reciprocal transactional activities. Such sophisticated interaction dynamics give rise to the emergence of cycle structures, as exemplified by the blue triangular motif in Fig. 1. Conventional GNNs are neural network implementations of the Weisfeilier-Leman (1-WL) graph isomorphism test algorithm, which leads to the expressive power of GNNs being within the bounds of 1-WL [16]. This limited expressive power is reflected in the ability to encode only subtree structures, but not subgraphs derived from arbitrary connectivity patterns, especially subgraphs with cycle structures [17]. This structural perception deficiency fails to capture transactions between neighbor nodes, allowing them to construct targeted transaction channels to evade regulation, as shown on the left side of Fig. 2. **2) Modeling of long-range dependencies.** As evidenced by Fig. 1, phishing accounts (colored red) spatially locate at distances beyond one hop, posing challenges for discriminative node representation learning. Ideally, exploring non-local topological aggregation of homophilic information facilitates the discriminability of node representations. For example, modeling long-range dependencies among all phishing accounts and aggregating global homophilic information. However, most existing GNNs [3], [18], [19], [20], [21] for Ethereum fraud detection typically adopt the popular 1-hop local aggregation paradigm to aggregate neighbor messages, with the receptive field bounded by the 1-WL test [22], [23], as shown on the left side of Fig. 2. This empirical evidence contradicts the heterophily of the Ethereum network [13], [24]. As illustrated in Fig. 1, the phenomenon of "opposites attract" can be observed, i.e., the features or classes of neighboring nodes are different. The local receptive field of GNNs makes it difficult to capture long-range dependencies between nodes with similar features for routing messages, ultimately leading to decision errors and performance degradation. **3) Selection of class-oriented information.** The real-world Ethereum network is inherently information-diverse, encompassing phishing accounts and various normal accounts, as shown in Fig. 1. From an information-theoretic perspective, selectively aggregating information from homophilic nodes in heterophilic graphs is conducive to enhancing the discriminability of node representations, thereby improving the performance of downstream tasks. However, existing graph machine learning methods have been empirically observed to struggle with separating class-oriented contexts due to the lack of a global attention perspective to filter task-irrelevant information and select task-oriented information [25]. This issue is particularly pronounced in large-scale Ethereum networks, where there is a requirement to focus on or ignore specific inputs amidst dense information.

To tackle these challenges, we design a structure-aware Mamba framework, named SAMamba, which learns the behavior patterns of Ethereum accounts from both micro and macro perspectives and selectively memorizes relevant information to achieve accurate fraud detection. As shown on the right side of Fig. 2, the core components of SAMmaba include the local structure preservation (LSP) module and the global information selection (GIS) module. Specifically, to address challenge (1), the LSP module uses local structured subgraphs combined with a local self-attention mechanism to encode local features, aiming to capture complex and diverse structural patterns. To address challenge (2), inspired by Mamba's exceptional long sequence modeling capabilities, the GIS module uses Mamba to model long-distance pairwise dependencies between nodes, aiming to achieve equivalent information exchange between all pairs of nodes. To address challenge (3), the GIS module is integrated with a selective state space model to perform input-dependent information filtering and adaptive context selection. Through in-depth exploration of structural patterns and adaptive information filtering, the SAMmaba framework provides a comprehensive understanding of transaction behavior, thereby significantly improving performance in fraud detection tasks. To assess the model's efficacy, SAMamba underwent rigorous testing on benchmark datasets EPTransNet and lw-AIG. Experimental evaluations corroborate that the model outperforms all comparative baselines, establishing its state-of-the-art performance in the target task. SAMmaba's exceptional performance provides reliable clues for tracking anonymous criminals, aiming to build security supervision tools for the Ethereum ecosystem.

Contribution. Our main contributions are summarized as follows.

1) This paper conducts an in-depth investigation into the capture of complex interaction patterns and long-range dependency modeling in the Ethereum network. The subgraph encoding strategy and selective state-space model are applied to structure-aware long-range

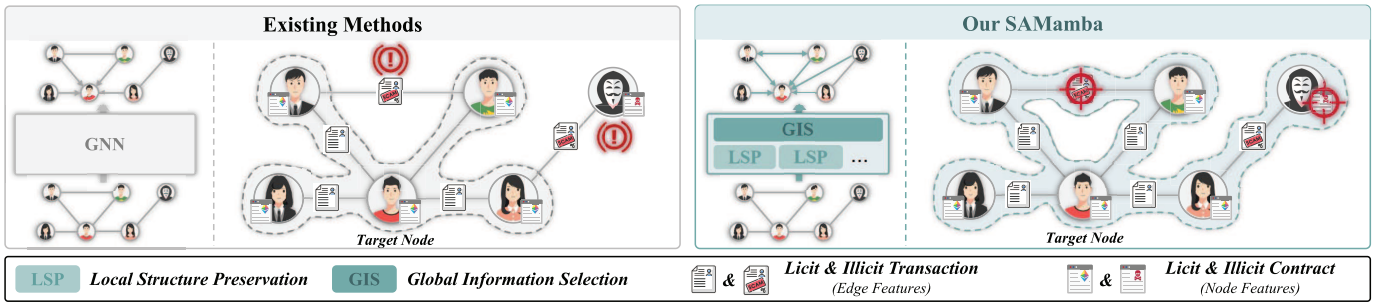


Fig. 2. Comparison of SAMamba with standard GNN-based methods. Existing methods use GNN's message passing mechanism similar to the 1-WL algorithm to encode the subtree structure around a target node, and a layer of GNN aggregates one-hop neighbors around the target node. SAMamba encodes a subgraph around a target node, allowing transactions between neighbors to be perceived, and broadens the receptive field to the full-graph perspective to gather information about high-order neighborhoods.

dependency modeling, aiming to enhance fraud detection for Ethereum.

2) A novel SAMamba framework, which integrates a local structure preservation module and a global information selection module. The former uses a subgraph encoding strategy to capture local complex interaction patterns, and the latter is combined as a powerful global information filtering and routing implementation.
3) Extensive experimental evaluations on the EPTransNet and lw-AIG datasets corroborate that SAMamba achieves significant performance gains over state-of-the-art baselines, furnishing robust capabilities for criminal activity tracking in Ethereum's operational ecosystem.

II. Related Work

This work relates to Ethereum fraud detection via graph analysis. In this section, we introduce two mainstream graph analysis methods for Ethereum fraud detection: graph embedding-based methods and graph neural network-based methods.

A. Graph Embedding-Based Methods

Graph embedding methods use algorithms such as random walks to generate node representations, and obtain descriptions of complex systems from natural data structures such as graphs. The node representations generated by this method can be combined with machine learning models to perform downstream tasks. Trans2Vec [26] is a graph embedding algorithm based on DeepWalk [27] and Node2Vec [28], which is specifically designed for the fraud detection task. It captures the relationship between accounts through random walks, collects transaction information to generate node representations, and combines support vector machines (SVM) to classify nodes into normal nodes and phishing nodes. However, Trans2Vec ignores the rich temporal information in dynamic transaction networks. To address this problem, other works [29], [30] aim to capture more comprehensive characteristics of dynamic transaction networks. These studies propose to model the Ethereum transaction network as a time-weighted multi-directional graph and design flexible time walk strategies for this large-scale network. The DeepWalk-based method has good expressiveness and strong performance in fraud detection tasks. However, these weakly generalized methods are difficult to generalize to unseen nodes and cannot easily incorporate node or edge attributes. In contrast, GNN-based methods are able to fuse graph topology properties to perform end-to-end tasks.

B. Graph Neural Network-Based Methods

The massive transaction data of Ethereum can be modeled as an account transaction graph. GNNs learn the representation of graph topology and node features through a series of message aggregation, and complete the tasks on the graph end-to-end. Researchers have applied graph neural network (GNN) models to this structured graph data in order to learn about anonymous accounts and enable account identification. In previous GNN-based fraud detection research [3], [18], the end-to-end graph convolutional network (GCN) model [31] has been used to map transaction patterns to account identity representations, enabling the identification of accounts as "normal", "phishing", or "bot" categories. In response to the excessive computation caused by a large amount of real transaction data, I^2BGNN adopts a sampling mechanism to limit the size of the transaction graph and implements identity reasoning based on end-to-end GCN. However, the GCN model has limitations in learning the correlations between interactive accounts. To address this issue, Ethident [19] and AEtransGAT [21] employ Graph Attention Networks (GAT) [32] to effectively characterize the correlations between accounts or transactions. However, bounded by the expressive power of subtree encoding strategies, current GNN-based methods still struggle to harness the complex behavior patterns in the Ethereum network. Existing GNN-based methods are still limited in terms of receptive field, which greatly impairs the performance of fraud detection tasks. In contrast, our SAMamba encodes sub-graph level representations and models long-range dependencies within the entire graph, effectively widening the receptive field to avoid the limitations of the aforementioned methods.

C. Subgraph Graph Neural Networks

Subgraph Graph Neural Networks, as an emerging paradigm in graph learning, have garnered significant attention. This paradigm employs Message Passing Neural Networks

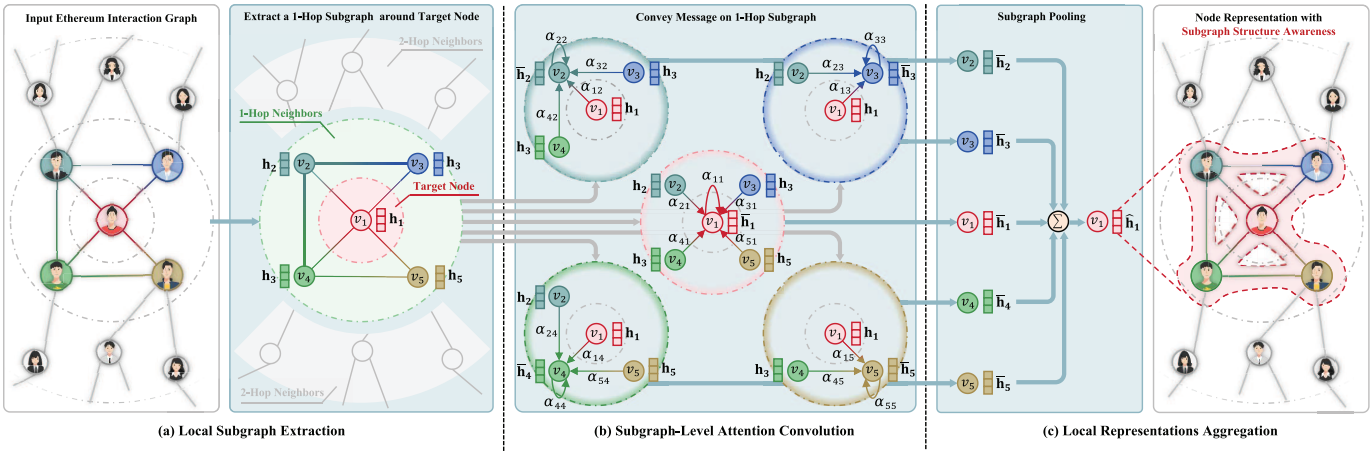


Fig. 3. The message-passing flow of the LSP follows three stages: (1) Extracting 1-hop subgraphs around the target node from the original graph; (2) Applying the local self-attention mechanism independently to each node in the subgraph; (3) Aggregating the node representations into a subgraph representation using subgraph pooling.

(MPNNs) on subgraph sets extracted from raw input graphs, followed by aggregation to derive subgraph-level representations with strengthened expressivity. ESAN [33] represents a graph as a set of subgraphs generated by node deletion, edge deletion, or ego-network strategies. It uses a Siamese module to process each subgraph independently and an information-sharing module to aggregate information from all subgraphs. GNN-AK [34] employs GNNs as subgraph encoders to compute node representations by encoding surrounding induced subgraphs, rather than merely encoding the direct neighbors of nodes. SUN [35] reconstructs the symmetry of subgraph sets to break through the upper bound of expressiveness based on existing subgraph GNNs, and demonstrates strong generalization ability by distinguishing update rules for root and non-root nodes. PSA-GNN [36] introduces a pair of parallel convolutional layers based on the bipartite graph between nodes and prior subgraphs, constructing a mixed receptive field by merging prior subgraphs into neighborhoods to further enhance the receptive field of subgraph GNNs. Despite the remarkable expressive power of subgraph GNNs, subgraph-level interaction patterns in Ethereum transaction scenarios remain underexplored. In particular, the complex linkages constructed by fraudulent transaction participants deserve dedicated preservation of their local structures to facilitate detection and forensics.

III. Method

A. Framework Overview

The Structure-Aware Mamba (SAMamba) framework is specifically designed to enhance the Ethereum fraud detection by taking advantage of structure preservation and feature selection. As illustrated in Fig. 2, SAMamba is a hierarchical framework that combines sophisticated attention mechanism and selection mechanism, and consists of two complete modules: the local structure preservation (LSP) module focuses on incorporating structural information with local attention mechanisms to emphasize subgraph-level locality. The Global Information Selection (GIS) module adopts a state space model to densely deliver messages within the global context window, combined with a selection mechanism to filter irrelevant information and remember important information indefinitely. This dual-module approach enables SAMamba to provide comprehensive analysis tools for Ethereum accounts, enabling Ethereum account detection and analysis from a dual perspective by performing context-sensitive key structure and feature selection.

B. Local Structure Preservation

The Local Structure Preserving (LSP) module is a key component of the SAMamba framework, which is specifically designed to enhance the capture of complex transaction patterns. It utilizes a complex message passing flow to aggregate more detailed local information to provide higher expressive power than GNN. GNN iteratively aggregates neighboring node features to a target node and learns node representations that encode their local structure and feature information [3], [11], [12], [13], [18], [19], [24]. However, the message passing paradigm of GNN encoding subtrees fails to capture complex substructures [17], [37], especially those with cycle. To address this challenge, the core idea of LSP is to encode the subgraph around the node instead of encoding the subtree. This insight is crucial for successfully discovering account identities, as these substructures allow capturing complex interactions, as shown in Fig. 3.

1) Local Subgraph Extraction: A graph G is a pair of (V, E) with a finite vertex set V and an edge set E, where $V = \{v_1, v_2, \ldots, v_n\}$ represents nodes and $E \subseteq \{(v_i, v_j) \in V \times V | v_i \neq v_j\}$ represents the edge. Each node $v_i \in V$ is associated with a feature vector $x_i \in X$, where $X \in \mathbb{R}^{n \times d}$ is a feature matrix, including nd-dimensional feature vectors. For $V[S] \subseteq V$, the graph $G[S] = (V[S], E[S])$ is a subgraph of G induced by the nodes $V[S]$, where $E[S] = \{(v_i, v_j) \in E | v_i, v_j \in V[S]\}$. The 1-hop neighbor of a node v_i in the subgraph $G[S]$ is denoted as $\mathcal{N}_{G[S]}(v_i) = \{v_j \in V[S] | (v_i, v_j) \in E[S]\}$. The extracted subgraph structure exhibits exceptional capability in characterizing cycle structures, a feature inherently absent in subtree structures. As illustrated in Fig. 3, the cycle structures

formed by node sets $\{v_1, v_2, v_3\}$, $\{v_1, v_2, v_4\}$, and $\{v_1, v_4, v_5\}$ are efficiently captured. This enhanced sensitivity to complex transaction patterns is of paramount importance for fraud detection.

2) Subgraph-Level Attention Convolution: To ensure that a node's representation encodes a local subgraph, the LSP module uses a sophisticated local self-attention mechanism to perform message passing within subgraphs. For each subgraph in S_G, this local self-attention mechanism facilitates message passing between a node and its neighbors in a way that preserves the structural integrity. Specifically, for any account node v_i in the input subgraph, the local attention mechanism learns the contribution of its neighbor v_j by calculating the attention scores as follows:

$$e_{ij}^{(l)} = \text{LeakyReLU}\left(\mathbf{W}_e^{(l)} \cdot \left[\mathbf{h}_i^{(l)} \| \mathbf{h}_j^{(l)}\right]\right), \tag{1}$$

where $\|$ denotes the concatenation operation, $\mathbf{h}_i^{(l)} \in \mathbf{H}^{(l)}$ denotes the hidden state vector of node v_i at the l-th layer, and $\mathbf{H}^{(l)} \in \mathbb{R}^{n \times d}$ denotes the hidden state matrix. After the linear transformation parameterized by $\mathbf{W}_e^{(l)}$, the nonlinear activation function LeakyRelu is executed to calculate the importance of account v_j to account v_i. At the very first, we have $\mathbf{h}_i^{(0)} = \mathbf{x}_i$. Additionally, to distinguish the attention scores between different accounts, the softmax function on neighboring accounts is used to further normalize the attention scores:

$$\alpha_{ij}^{(l)} = \text{Softmax}\left(e_{ij}^{(l)}\right) = \frac{\exp\left(e_{ij}^{(l)}\right)}{\sum_{v_x \in \mathcal{N}_{G[S]}(v_i) \cup \{v_i\}} \exp\left(e_{ix}^{(l)}\right)}. \tag{2}$$

Once the normalized attention scores are obtained, the neighborhood context is aggregated to update the features of the target node:

$$\bar{\mathbf{h}}_i^{(l)} = \text{ReLU}\left(\alpha_{ii}^{(l)} \cdot \mathbf{W}_\alpha^{(l)} \cdot \mathbf{h}_i^{(l)} + \sum_{v_j \in \mathcal{N}_{G[S]}(v_i)} \alpha_{ij}^{(l)} \cdot \mathbf{W}_\alpha^{(l)} \cdot \mathbf{h}_j^{(l)}\right), \tag{3}$$

where a linear transformation parameterized by $\mathbf{W}_\alpha^{(l)}$ and a non-linear activation function Relu are applied to produce the hidden representation $\bar{\mathbf{h}}_i^{(l)}$. It's worth noting that the local node representations are aggregated in a position-sensitive manner from the representations of neighboring nodes $v_j \in \mathcal{N}(v_i)$, ensuring effective capture of the local structural relationships.

3) Local Representations Aggregation: A pooling layer is used to aggregate the representations of all nodes within the subgraph into a single subgraph representation, which is then used as the representation of the target node. Such processing not only presvers the structural information of the subgraph, but also effectively integrates the features of each node to construct a comprehensive target node representation. Subsequently, the original features of the target node are enhanced by concatenating them with the subgraph representation, resulting in the updated representation of the target node v_u.

$$\hat{\mathbf{h}}_u^{(l)} = \mathbf{h}_u^{(l)} + \sum_{v_i \in \mathcal{N}_{G[S]}(v_u) \cup \{v_u\}} \bar{\mathbf{h}}_i^{(l)}. \tag{4}$$

C. Global Information Selection

The Global Information Selection (GIS) module is designed to model long-range interactions of Ethereum accounts from a global perspective. Some intrinsic limitations of GNNs, including over-smoothing [38], over-squeezing [39], [40], and poor capture of long-range dependencies [41], [42], greatly impair their applicability to heterophilic and long-range graphs. Graph Transformer (GT) [43], [44], [45] thus serve as an alternative to GNNs. However, GT is easily distracted by irrelevant context when densely routing global information [46]. Addressing this challenge, the GIS module leverages Mamba's powerful information filtering capabilities to incorporate important information from the global window, thereby enhancing the sniffing of critical information in the Ethereum network. The message-passing flow of GIS as illustrated in Fig. 4.

1) Subgraph-Based Node Prioritization: The Mamba architecture is designed for sequential data, where each node is updated based on hidden state derived from the prior node. Nodes fed into Mamba earlier capture less information about the context of the sequence, while nodes entered later capture information about almost the entire sequence. Therefore, the sequence needs to be sorted according to the importance of the nodes. As shown in Fig. 4(a), the nodes on the graph are flattened into the sequence $\hat{\mathbf{H}}^{(l)} = \{\hat{\mathbf{h}}_i^{(l)} | v_i \in V\}$, without any specific order. Next, the node sequence is sorted in ascending order:

$$\mathbf{H}_{\text{sorted}}^{(l)} = \text{NodeSort}(\hat{\mathbf{H}}^{(l)}), \tag{5}$$

where NodeSort$(\cdot)$ is a flexible node priority strategy. This strategy first counts the sum of the degrees of all nodes on the subgraph corresponding to the input node, and input nodes are sorted in ascending order according to the corresponding sum of degrees. Intuitively, the subgraph structure contains more information than the subtree structure, and its outstanding expressive power shows its importance in downstream tasks. This insight justifies ordering nodes based on the total degree of all nodes on a subgraph.

2) Long-Range Interaction Modeling and Selection: The State-Space Model (SSM) is employed to learn long-range dependencies for expressive graph summarization. Specifically, the sorted node sequence are normalized and fed to the gating unit and 1-D convolution respectively:

$$\begin{aligned}
\mathbf{H}_{\text{norm}}^{(l)} &= \text{LayerNorm}(\mathbf{H}_{\text{sorted}}^{(l)}), \\
\mathbf{H}_{\text{gate}}^{(l)} &= \text{SiLU}(\mathbf{W}_{\text{gate}}^{(l)} \mathbf{H}_{\text{norm}}^{(l)}), \\
\mathbf{H}_{\text{input}}^{(l)} &= \text{SiLU}(\text{Conv}(\mathbf{W}_{\text{input}}^{(l)} \mathbf{H}_{\text{norm}}^{(l)})),
\end{aligned} \tag{6}$$

where $\mathbf{W}_{\text{gate}}^{(l)}$ and $\mathbf{W}_{\text{input}}^{(l)}$ are learnable parameters. To adaptively select relevant information from the context, Mamba stands out by implementing a selectivity mechanism as its core SSM operator. This mechanism focuses on making the parameters that affect sequence interactions dependent on the input:

$$\begin{aligned}
\mathbf{B}^{(l)} &= \mathbf{W}_{\mathbf{B}}^{(l)} \mathbf{H}_{\text{input}}^{(l)}, \\
\mathbf{C}^{(l)} &= \mathbf{W}_{\mathbf{C}}^{(l)} \mathbf{H}_{\text{input}}^{(l)}, \\
\Delta^{(l)} &= \text{Softplus}(\mathbf{W}_{\Delta}^{(l)} \mathbf{H}_{\text{input}}^{(l)}),
\end{aligned} \tag{7}$$

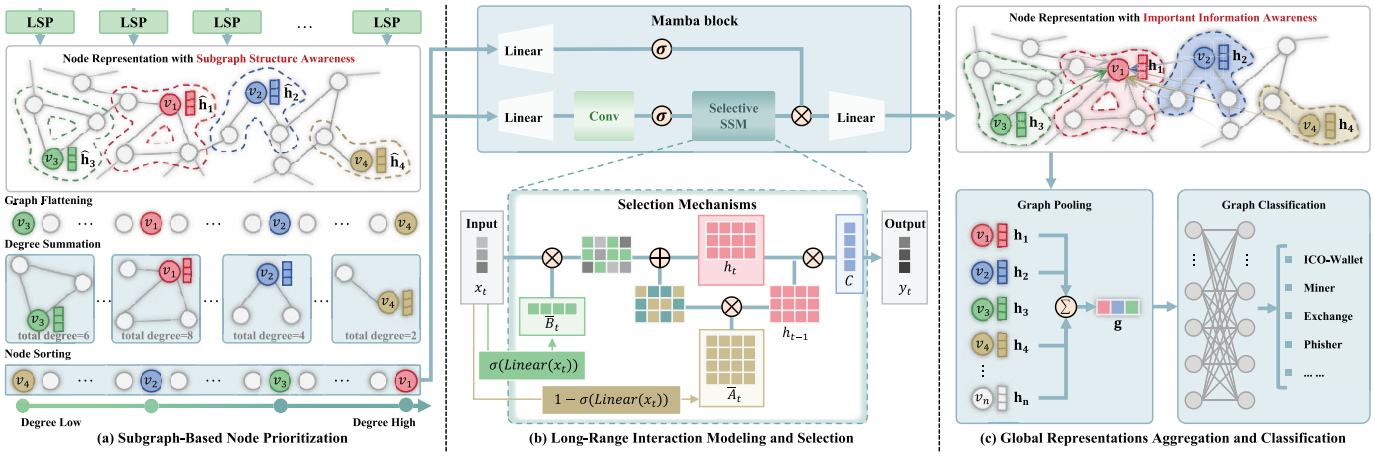


Fig. 4. The message-passing flow of GIS follows three stages: (1) LSP extracts all node representations on the entire graph and sorts them based on the degree of the nodes; (2) Mamba is introduced to model long-distance dependencies and adaptively filters and aggregates global information through a selection mechanism; (3) Graph pooling is used to aggregate the final node representation into a graph representation, and MLP is combined to classify the graph representation.

where $\mathbf{B}^{(l)} \in \mathbb{R}^{N\times 1}$, $\mathbf{C}^{(l)} \in \mathbb{R}^{N\times 1}$ denotes the input matrix and output matrix of SSM respectively, and $\Delta^{(l)}$ denotes the time scale parameter. In the Equation (7), the parameters $\mathbf{B}^{(l)}$, $\mathbf{C}^{(l)}$ and $\Delta^{(l)}$ are directly derived from the input data $\mathbf{H}^{(l)}_{\text{input}}$, thus giving Mamba the intrinsic ability of context sensitivity and adaptive weight modulation. Next, the discretization process is performed:

$$\begin{aligned}\bar{\mathbf{A}}^{(l)} &= \text{Discrete}_{\mathbf{A}}(\mathbf{A}^{(l)}, \Delta^{(l)}),\\ \bar{\mathbf{B}}^{(l)} &= \text{Discrete}_{\mathbf{B}}(\mathbf{B}^{(l)}, \Delta^{(l)}),\end{aligned} \tag{8}$$

where $\mathbf{A} \in \mathbb{R}^{N\times N}$ denotes the evolution matrix of SSM, and Discrete$(\cdot)$ denotes discrete transformation, usually using zero-order hold (ZOH) method, which is defined as follows:

$$\begin{aligned}\bar{\mathbf{A}} &= \exp(\Delta\mathbf{A}),\\ \bar{\mathbf{B}} &= (\Delta\mathbf{A})^{-1}(\exp(\Delta\mathbf{A} - \mathbf{I})) \cdot \Delta\mathbf{B}.\end{aligned} \tag{9}$$

In Equation (9), the input data dependence of Δ makes $\bar{\mathbf{A}}$ also related to the input data according to $\bar{\mathbf{A}} = \exp(\Delta\mathbf{A})$. Next, SSM is employed to model long-distance dependencies and route information between nodes:

$$\mathbf{H}^{(l)}_{\text{output}} = \text{SSM}(\bar{\mathbf{A}}^{(l)}, \bar{\mathbf{B}}^{(l)}, \mathbf{C}^{(l)})(\mathbf{H}^{(l)}_{\text{input}}), \tag{10}$$

where SSM$(\cdot)$ represents the state space model, which is called a linear time-invariant system that maps the input sequence. Formally, SSM models the input data using an ordinary differential equation (ODE), which transforms the input sequence $x(t) \in \mathbb{R}^N$ to the output sequence $y(t) \in \mathbb{R}^N$ through a hidden state $h(t) \in \mathbb{R}^N$:

$$\begin{aligned}h'(t) &= \mathbf{A}h(t) + \mathbf{B}x(t),\\ y(t) &= \mathbf{C}h(t),\end{aligned} \tag{11}$$

where the output sequence $y(t)$ at time t is obtained by calculating $h(t)$, but $h(t)$ is difficult to solve analytically in a deep learning environment. In contrast, real-world data is often discrete rather than continuous. Therefore, this continuous ODE is approximated by the discretization process in Equation (9), where the time scale parameter Δ converts the continuous parameters $\mathbf{A}$, $\mathbf{B}$ into discrete parameters $\bar{\mathbf{A}}$, $\bar{\mathbf{B}}$, which are defined as follows:

$$\begin{aligned}h_t &= \bar{\mathbf{A}}h_{t-1} + \bar{\mathbf{B}}x_t,\\ y_t &= \mathbf{C}h_t,\end{aligned} \tag{12}$$

In the Equation (12), due to the data dependence of $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$, and reviewing the main principles of Mamba, $g_t = \sigma(\text{Linear}(\text{x}_\text{t}))$ and $h_t = (1-g_t)h_{t-1} + g_t x_t = \bar{\mathbf{A}}h_{t-1} + \bar{\mathbf{B}}x_t$. When the current input x_t updates the hidden state h_t, g_t controls the balance between the current input x_t and the previous context h_{t-1}, thereby filtering node representations that are irrelevant to downstream tasks based on the input data. Next, the output of SSM $\mathbf{H}^{(l+1)}$ is gated by the projection of the original input $\mathbf{H}_{\text{gate}}$:

$$\mathbf{H}^{(l+1)} = \mathbf{W}_{\text{output}}(\mathbf{H}_{\text{output}} \odot \mathbf{H}_{\text{gate}}), \tag{13}$$

where $\odot$ denotes element multiplication, which means $\mathbf{H}_{\text{output}}$ is gated by another representation $\mathbf{H}_{\text{gate}}$.

3) Global Representations Aggregation and Classification: For graph classification tasks, a graph pooling layer summarizes the final node representations $\mathbf{h}^{(l+1)}_u \in \mathbf{H}^{(l+1)}$ into a graph representation. Each graph representation corresponds to an account representation, and this enhanced account representation integrates insights into local interaction details and an understanding of global broad information:

$$\mathbf{g} = \sum_{v_u \in V} \mathbf{h}^{(l+1)}_u. \tag{14}$$

For node classification tasks, the above steps are omitted. Finally, we take the graph representation $\mathbf{g}$ as the input to an MLP layer with a softmax function:

$$\hat{y} = \text{Softmax}(\text{MLP}(\mathbf{g})). \tag{15}$$

The advantage of the SAMamba framework of stacking GIS modules on top of LSP modules is intuitive: on the one hand, LSP serves as a specialized architecture for learning local representations of the structure of a node's immediate neighborhood, which explicitly encodes structural relationships to

capture transaction patterns. On the other hand, GIS functions as a powerful global reasoning module, facilitates node filtering between global connections to select more important transaction patterns or account features. By rationally integrating the LSP module and the GIS module, the SAMamba framework shows excellent performance in downstream tasks such as Ethereum fraud detection, as shown in Table V.

D. Loss Function

For the purpose of Ethereum account identification, SAMamba is trained using the cross entropy function, which is given by:

$$\mathcal{L}_{\mathrm{p}} = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\left(\hat{y}_i\right), \tag{16}$$

where $\mathcal{L}_{\mathrm{p}}$ is the cross entropy loss, N represents the number of samples in a batch.

To address the potential data scarcity issue and enhance feature extraction capabilities, we introduce GraphCL [47], which employs two augmented views to jointly train an encoder. The pre-training is achieved by maximizing the consistency between the representations of the two augmented views of the same graph in the latent space through a contrastive loss:

$$\mathcal{L}_i = -\log \frac{e^{\mathrm{sim}\left(\mathbf{z}_i^a, \mathbf{z}_i^b\right)/\tau}}{\sum_{j=1, j\neq i}^{N} e^{\mathrm{sim}\left(\mathbf{z}_i^a, \mathbf{z}_j^b\right)/\tau}}, \tag{17}$$

where τ is the temperature parameter, $\mathbf{z}_i^a$ and $\mathbf{z}_i^b$ represent two augmented graph representations, which are generated according to GraphCL [47] by dropping nodes from the i-th graph in a batch of N samples. sim$(\cdot)$ represents the cosine similarity function, defined as $\mathrm{sim}\left(\mathbf{z}_i^a, \mathbf{z}_i^b\right) = \mathbf{z}_i^{a\top}\mathbf{z}_i^b / \left\|\mathbf{z}_i^a\right\| \left\|\mathbf{z}_i^b\right\|$. The objective of graph-level contrast is to maximize the consistency between two related graph-enhanced views in the contrastive space by minimizing the contrastive loss, yielding a self-supervised loss:

$$\mathcal{L}_{\mathrm{s}} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_i. \tag{18}$$

The complete loss function consists of two parts: the self-supervised loss function and the classification task loss function:

$$\mathcal{L} = \mathcal{L}_{\mathrm{p}} + \lambda \cdot \mathcal{L}_{\mathrm{s}}, \tag{19}$$

where λ is a balancing hyperparameter that controls the contribution of the self-supervised loss function.

IV. Experiments

In this section, we conduct a series of experiments to evaluate the performance of the proposed method on the Ethereum account interaction graph dataset. We compare the achieved results with other state-of-the-art methods in this field and give a detailed analysis of the experimental results to demonstrate the effectiveness of the proposed SAMamba.

TABLE I

Statistics of Lw-AIG Dataset. $|G|$ is the Number of Graphs, Avg. $|N|$ and Avg. $|E|$ are the Average Number of Nodes and Edges in Graphs Respectively, $|x|$ and $|e|$ are the Number of Node and Edge Features

Subdataset	Categories	$\lvert G\rvert$	Avg.$\lvert N\rvert$	Avg.$\lvert E\rvert$	$\lvert x\rvert$	$\lvert e\rvert$
Eth-ICO	Amount	146	42.5	141.3	14885	2
	Times		52.2	152.6		
	avgAmount		42.4	140.7		
Eth-Mining	Amount	130	23.7	72.9	14885	2
	Times		24.7	67.2		
	avgAmount		29.0	91.9		
Eth-Exchange	Amount	386	33.6	123.7	14885	2
	Times		38.0	113.4		
	avgAmount		38.6	148.6		
Eth-Phish&Hack	Amount	5070	37.3	110.8	14885	2
	Times		37.8	101.6		
	avgAmount		37.8	111.3		

A. Dataset

1) Lw-AIG: We evaluate SAMamba on a real Ethereum transaction dataset released on the Xblock platform [48]. The dataset is composed of an account relationship interaction graph, constructed from Ethereum user relationships through interaction merging and feature construction operations to form a lightweight account interaction graph (lw-AIG) [19]. In lw-AIG, nodes represent accounts, and edges represent transaction data and contract callbacks. The dataset is divided into four subdatasets based on account identity labels from the Ethereum blockchain explorer's Label Word Cloud: ICO wallets, mining accounts, exchange accounts, and phishing&hack accounts. ICO wallets are characterized by a high number of outbound edges distributing investment returns from the central ICO account to surrounding supporters; mining accounts show a pattern of numerous outbound edges from a central pool to peripheral mining nodes with accumulated rewards; exchange accounts, interacting frequently with clients, appear as highly central hub nodes with significant in-degree and out-degree; phishing&hack accounts are distinguished by few bidirectional edges (mutual transactions) between the central node and surrounding nodes, featuring high in-degree and low out-degree for the central node. For each subdataset, graph sampling based on different edge information (Amount, Times, or avgAmount) generates three categories of graph, as shown in Table I. The ratio of sampled positive to negative samples is maintained at 1:1 to ensure data balance. Each graph corresponds to a target account, forming a dataset: $D = \{(g_i, y_i)|\forall(v_i, y_i) \in Y\}$. The labels of target accounts are assigned to the graphs, for learning a function that maps graph patterns to account identity labels. The complex interaction patterns in the lw-AIG dataset lay the foundation for exploring the structure-aware capabilities of the LSP module, demonstrating SAMamba's ability to capture various complex fraudulent behaviors.

2) EPTransNet: EPTransNet represents a real-world Ethereum transaction network comprising phishing accounts

and normal accounts. The original large-scale graph consists of 2,973,382 nodes and 13,551,214 edges, with 1,157 labeled phishing nodes–exhibiting severe class imbalance between positive and negative samples. Adhering to the strategy outlined in [3], we employ random walk sampling to generate graphs of sizes 30,000, 40,000, and 50,000 nodes, respectively. Node feature engineering follows [3]. The objective of EPTransNet is to detect phishing accounts within massive account datasets, constituting a node-level classification task with imbalanced class labels. This large-scale graph provides a reliable scenario for evaluating the long-range dependency modeling capability of the GIS module and offers insights into the rationality of SAMamba's systematic design. Statistical summaries of the EPTransNet dataset are presented in Table IV.

TABLE II
STATISTICS OF EPTRANSNET DATASET. $|L|$ IS THE NUMBER OF NODES LABELED AS PHISHING ACCOUNTS. $|E|$ IS THE NUMBER OF EDGES. AVG.$|D|$ IS THE AVERAGE DEGREE. $|x|$ AND $|e|$ ARE THE NUMBER OF NODE AND EDGE FEATURES

Subdataset	$\|L\|$	$\|E\|$	Avg.$\|D\|$	$\|x\|$	$\|e\|$
30,000	125	1,268,798	84.58	8	2
40,000	163	1,374,388	68.71	8	2
50,000	197	1,566,301	62.65	8	2

B. Comparison Methods

To evaluate the effectiveness of our method, we compare three types of baselines in the experiment: graph embedding-based methods, GNN-based methods and GT-based methods. The graph embedding methods employ techniques such as DeepWalk [27], Node2Vec [28], and Trans2Vec [26] to encode accounts into vector-space representations. These account embeddings are subsequently fed into two machine learning classifiers: Logistic Regression (LR) and Random Forest (RF) to accomplish the account identification task. For the GNN-based methods, we conduct comparative analyses between our method and specialized fraud detection methods, encompassing I^2BGNN [18], Ethident [19], FAGNN [20], and AEtransGAT [21]. Notably, I^2BGNN executes account identification by capitalizing on different edge information and presents two variants: I^2BGNN-A and I^2BGNN-T. Additionally, we compared the proposed method with Graph Transformer methods, including SGFormer [49] and GraphGPS [45]. SGFormer is architected to model interdependencies among massive nodes in large-scale graphs, thereby enabling comparative assessment of long-range dependency modeling capabilities. For GraphGPS, the evaluation focal point was anchored in the selection of global attention modules within its framework, with performance assessed via implementations of dense attention (Transformer [50]) and diverse sparse attention mechanisms (Performer [51]). For the lw-AIG dataset, to adapt these models to graph classification tasks, we integrated supplementary graph pooling layers and prediction heads. In the case of EPTransNet, graph pooling layers were eliminated to focus on node-level classification.

C. Implementation Details

Our proposed SAMamba, is implemented using the PyTorch framework and trained on an NVIDIA Tesla V100S-PCIE-32GB GPU. During the training process, we use the following hyperparameters: batch size of 64, initial learning rate of 0.0003, 10 rounds of linear warm-up, cosine annealing learning rate scheduler, and AdamW optimizer with a weight decay of 0.0001. We train the model for 1000 epochs and apply early stopping with a patience of 20. K-fold cross validation is used to evaluate model performance on unseen data. We randomly split the dataset into k mutually exclusive subsets of equal size, and then use k-1 subsets as training sets each time and the remaining subset as test set. To report the final results, we repeat the 3-fold cross-validation procedure 3 times with different random seeds and calculate the mean and standard deviation of the evaluation metrics. In the EPTransNet dataset, the minority class (i.e., phishing accounts) is of greater importance. Therefore, we trained all models using a weighted cross-entropy loss with a weight ratio of 1:9 for normal to phishing account samples, thereby enhancing the significance of phishing accounts. Furthermore, considering the ample data volume available for EPTransNet, graph contrastive learning [47] was excluded from the experiments on this dataset to alleviate additional computational overhead.

D. Evaluation Metrics

The graph classification and node classification tasks related to the lw-AIG dataset and the EPTransNet dataset can be described as binary classification problems. Due to the balanced positive and negative samples in the lw-AIG dataset, we use the Micro-F1 score as the main evaluation metric. Micro-F1 emphasizes the overall performance of each category to comprehensively evaluate detection performance by assigning equal weights to all category samples, which is particularly suitable for comparing the generalization capabilities of models in sample-balanced scenarios. For the EPTransNet dataset with severely unbalanced phishing and normal accounts, we introduce Recall, Precision, and F1 score as evaluation metrics. Recall refers to the proportion of positive samples (phishing accounts) correctly identified by the model to all actual positive samples. Precision indicates the proportion of correctly identified positive samples to all samples predicted as positive. As a comprehensive measure of Precision and Recall, the F1 score balances the weights of the two through a harmonic mean formula.

E. Classification Performance

We conduct a comprehensive comparison of our proposed SAMamba model against various baseline methods, and the results are presented in Table III. SAMamba demonstrates remarkable superiority over all graph embedding baselines in terms of Micro-F1 scores across four sub-datasets. This performance boost can be attributed to the ability of our SAMamba framework to capture the behavioral patterns of different account types when extracting subgraph-level features. Compared to GNN-based methods, SAMamba demonstrates

TABLE III

SUMMARY OF CLASSIFICATION PERFORMANCE IN TERMS OF MICRO F1-SCORE ON THE LW-AIG DATASET. THE HIGHEST PERFORMANCE IS MARKED IN BOLD, AND THE SECOND BEST PERFORMANCE IS UNDERLINED

Method	lw-AIG Dataset (with different sampling strategies)											
	Eth-ICO			Eth-Mining			Eth-Exchange			Eth-Phish&Hack		
	Amount	Times	avgAmount	Amount	Times	avgAmount	Amount	Times	avgAmount	Amount	Times	avgAmount
DeepWalk + LR	79.24±0.85	82.17±0.56	77.18±0.33	62.06±0.36	56.72±1.47	65.64±0.96	62.08±0.44	67.01±0.85	63.03±1.29	75.15±0.15	76.86±0.16	74.82±0.27
DeepWalk + RF	77.18±1.41	81.28±0.85	77.63±1.41	56.15±2.88	56.43±2.20	63.83±2.29	63.20±0.77	66.40±1.04	63.38±1.61	77.04±0.12	78.03±0.06	76.08±0.40
Trans2Vec + LR	59.83±4.49	65.11±0.53	66.22±1.61	55.12±7.15	50.77±4.90	56.12±0.65	56.90±1.27	55.35±2.25	57.85±3.21	63.49±3.71	61.07±2.77	56.43±1.40
Trans2Vec + RF	70.35±2.76	70.35±1.15	68.77±2.25	54.87±5.26	56.11±1.25	57.20±3.45	59.24±2.76	61.66±0.56	59.32±1.65	61.94±1.70	65.00±4.09	68.96±2.47
Node2Vec + LR	79.46±0.55	82.19±0.56	78.56±0.32	61.30±0.38	55.16±2.51	64.12±3.19	64.16±0.53	68.38±0.73	67.00±1.20	73.91±0.06	75.34±0.36	73.34±0.45
Node2Vec + RF	76.48±0.31	81.96±0.85	77.18±1.70	59.50±0.76	57.45±2.20	57.95±3.61	61.74±1.22	64.50±1.73	62.94±0.97	74.38±0.14	74.50±0.22	73.57±0.15
I^2BGNN-A	76.65±12.13	90.83±2.31	80.49±5.11	69.71±10.68	74.77±6.78	71.48±3.99	91.05±0.64	91.33±0.49	91.25±0.64	56.21±9.19	60.97±3.31	82.92±1.40
I^2BGNN-T	79.55±5.98	50.92±6.52	78.67±8.79	81.33±0.95	75.79±0.65	68.93±15.14	90.86±0.67	90.81±0.32	91.01±0.32	88.92±0.70	63.19±0.16	85.18±1.91
SGFormer	92.89±0.32	86.86±10.25	92.55±0.65	69.60±0.72	67.57±4.03	66.15±5.37	89.91±0.32	**92.83±2.50**	89.85±0.53	92.35±0.20	94.16±0.14	93.85±0.00
GPS-Transformer	82.87±11.25	72.20±9.20	83.68±6.26	67.41±3.58	71.62±3.51	70.90±1.61	87.76±0.55	62.79±4.94	60.59±5.37	81.81±0.25	83.28±1.81	82.09±5.14
GPS-Performer	76.14±4.28	72.57±9.71	83.86±4.58	61.07±7.53	72.86±8.47	73.90±8.64	63.38±2.57	66.91±3.43	87.94±0.74	73.63±4.54	81.44±4.89	73.80±1.36
Ethident	91.22±2.44	89.57±3.14	88.71±0.64	78.96±2.50	84.21±0.74	80.14±0.95	**92.32±1.38**	90.58±0.48	**92.61±0.55**	97.17±0.16	96.81±0.10	97.15±0.13
FAGNN	91.12±1.48	89.31±0.65	91.03±0.34	75.53±4.61	67.66±4.99	76.04±5.75	90.68±0.37	91.63±0.76	90.21±0.21	96.41±0.00	96.90±0.04	96.56±0.06
AEtransGAT	72.71±6.95	75.97±12.12	77.79±10.52	60.05±5.85	56.08±6.13	58.98±5.31	81.49±6.23	92.70±0.56	91.94±0.32	97.25±0.03	96.64±0.10	81.45±7.75
SAMamba	**93.25±1.71**	**92.29±2.32**	**92.97±1.17**	**86.16±0.96**	**86.34±1.37**	**85.12±1.92**	91.97±1.61	92.75±1.04	91.56±0.37	**97.44±0.15**	**97.09±0.13**	**97.45±0.21**

considerable improvements on three sub-datasets: Eth-ICO, Eth-Mining, and Eth-Phish&Hack, with Micro-F1 scores ranging from 0.19% to 4.98% higher than the best baselines. The results in Table III indicate that graph embedding methods perform significantly worse than GNN-based approaches, ranking relatively lower. This is because graph embedding methods cannot learn task-relevant features in an end-to-end manner and rely heavily on the selected classifier to achieve high performance. In contrast, GNN-based methods can learn from the topological structure and potential features of the graph, thus obtaining more informative representations of the account sub-graphs, leading to better performance. However, the GNN-based methods still have limitations, such as difficulties in learning important edge information, overly simplistic subtree structures, and insufficient semantic expression. In comparison, our SAMamba model extracts the subgraph of each node for information aggregation, enabling the acquisition of subgraph-level node representations.

For the EPTransNet dataset, considering the extreme class imbalance between phishing accounts and normal accounts, recall is prioritized to ensure the detection of all potential malicious accounts, thereby minimizing the risk of criminal activities evading identification due to false negatives. SGFormer and GPS-Performer, which incorporate a global attention mechanism, exhibit a remarkable superiority over Ethident, I2BGNN, FAGNN, and AEtransGAT that operate solely from a local-scope perspective, as shown in Table IV. This finding is in accordance with our anticipations. GNN architectures characterized by local receptive fields are inherently limited in establishing long-range dependencies on extensive graphs, thereby impeding the efficient routing of equivalent information. Conversely, GT-based methods harness the global attention paradigm to orchestrate the aggregation of equivalent messages, while effectively filtering out non-equivalent ones. Compared with the GT-based methods, SAMamba has achieved an improvement ranging from 1.95% to 14.06% in terms of recall, and an improvement ranging from 0.38% to 6.54% in terms of F1 score. This enhancement originates from SAMamba's integration of the capability to capture complex local patterns, a functionality conspicuously absent in traditional GT-based approaches.

F. Efficiency Performance

Table VI illustrates the computational costs on the Eth-Phish&Hack (Amount) subdataset. SAMamba achieves a FLOPs count of 0.09738912 GB, outperforming methods such as SGFormer, GPSPerformer, FAGNN, and AEtransGAT in computational efficiency. With 1.09 MB of parameters, SAMamba strikes an balance between model complexity and performance. Its GPU memory footprint of 42.57 MB remains within a reasonable range, ensuring efficient computation while imposing hardware requirements. Notably, SAMamba introduces a global perspective under linear complexity constraints—a capability absent in I2BGNNA, Ethident, FAGNN, and AEtransGAT. Simultaneously, it retains a locally complex receptive field that surpasses the capabilities of SGFormer, GPSPerformer, and GPSTransformer. Overall, SAMamba demonstrates excellent balance and efficiency in computational resource utilization and classification performance, as shown in Tables III and IV.

G. Ablation Studies

1) Evaluating the Effectiveness of Structural Subgraph: We systematically analyzed the impact of different encoding strategies on account classification performance, and the experimental results are shown in Table V. It is worth noting that there is no obvious performance difference when the

TABLE IV

SUMMARY OF CLASSIFICATION PERFORMANCE ON THE EPTRANSNET DATASET, INCLUDING PRECISION, RECALL AND F1 SCORE. THE HIGHEST PERFORMANCE IS MARKED IN BOLD, AND THE SECOND BEST PERFORMANCE IS UNDERLINED. OOM STANDS FOR OUT OF MEMORY

Method	EPTransNet Dataset (with different sampled graph sizes)								
	Graph Size=30,000			Graph Size=40,000			Graph Size=50,000		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
DeepWalk + LR	4.04±1.52	1.27±0.57	1.93±0.84	1.08±1.52	0.44±0.63	0.63±0.89	2.31±1.73	0.42±0.34	0.69±0.53
DeepWalk + RF	11.11±15.71	0.35±0.49	0.67±0.95	22.22±15.71	0.58±0.44	1.13±0.85	**33.34±12.00**	1.03±0.45	1.98±0.86
Trans2Vec + LR	3.61±0.91	2.13±0.40	2.66±0.55	5.66±2.08	1.65±0.21	2.43±0.36	2.82±1.99	0.69±0.62	1.05±0.88
Trans2Vec + RF	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	11.11±7.85	0.40±0.28	0.77±0.55
Node2Vec + LR	6.70±1.20	4.52±0.59	5.28±0.70	6.78±2.14	3.43±3.23	3.78±2.70	0.00±0.00	0.00±0.00	0.00±0.00
Node2Vec + RF	0.00±0.00	0.00±0.00	0.00±0.00	11.11±15.71	0.22±0.32	0.44±0.62	33.33±23.57	0.78±0.59	1.53±1.15
I^2BGNN-A	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
I^2BGNN-T	21.39±15.46	15.87±14.15	6.02±1.54	11.03±6.03	7.36±1.66	7.52±1.67	31.07±18.32	9.77±4.69	7.39±0.32
SGFormer	**26.73±3.76**	26.39±1.15	26.20±2.30	**29.43±10.25**	25.90±3.46	20.68±3.84	26.20±10.21	29.19±9.86	19.79±6.09
GPS-Transformer	0.05±0.07	11.11±15.71	0.10±0.14	0.00±0.00	0.00±0.00	0.00±0.00	OOM	OOM	OOM
GPS-Performer	19.77±2.74	24.89±2.21	20.96±1.67	16.97±9.43	19.44±4.25	16.18±6.22	23.05±1.64	21.69±0.73	22.04±1.01
Ethident	12.95±1.38	14.42±1.85	13.40±1.20	14.38±1.51	12.89±2.61	12.70±1.50	13.57±3.71	18.09±6.54	15.02±4.46
FAGNN	3.09±0.73	15.21±1.73	5.14±0.45	7.69±1.23	1.92±1.07	3.07±0.69	2.42±0.21	9.09±0.64	3.82±1.35
AEtransGAT	7.48±1.29	21.73±0.91	8.36±2.82	12.82±0.51	9.61±1.15	10.98±1.09	1.63±0.93	6.49±1.35	2.61±1.78
SAMamba	25.63±2.26	**28.34±1.51**	**26.58±1.93**	21.28±2.16	**39.96±5.71**	**27.22±2.13**	20.84±2.29	**31.47±5.52**	**24.40±0.91**

TABLE V

ABLATION STUDY OF CLASSIFICATION PERFORMANCE IN TERMS OF MICRO-F1 SCORE ON THE LW-AIG DATASET

Method		lw-AIG Dataset (with different sampling strategy)											
		Eth-ICO			Eth-Mining			Eth-Exchange			Eth-Phish&Hack		
		Amount	Times	avgAmount	Amount	Times	avgAmount	Amount	Times	avgAmount	Amount	Times	avgAmount
GIS		91.41±1.61	90.69±2.60	90.59±1.55	83.53±1.31	84.17±1.79	84.27±1.86	91.16±1.00	90.55±1.15	90.07±0.69	94.98±0.28	95.32±0.52	95.72±0.45
LSP (w/o LA)	subtree	91.82±1.69	91.61±2.58	91.88±1.16	84.52±1.22	84.28±1.38	84.82±1.91	90.15±0.71	90.33±1.17	89.85±0.97	95.53±0.43	95.44±0.90	95.82±0.35
	subgraph	91.95±1.66	90.79±2.30	90.85±1.29	84.20±1.44	85.24±1.22	84.38±2.05	90.57±0.59	90.47±1.23	90.04±0.82	95.69±0.47	95.52±0.42	96.15±0.32
LSP	subtree	91.58±1.61	90.75±2.19	90.76±1.28	83.56±1.38	84.01±1.17	84.92±1.61	90.52±0.74	90.42±0.98	90.01±0.97	94.97±0.59	94.87±0.27	95.64±0.73
	subgraph	91.16±1.55	90.27±2.12	92.01±1.13	84.45±1.39	83.99±1.45	85.03±1.60	90.12±0.96	90.47±1.03	90.18±0.88	95.77±0.39	95.42±0.32	95.87±0.35
SAMamba (w/o LA)	subtree	91.56±1.61	91.55±1.19	91.71±1.23	84.97±1.27	85.06±1.10	84.99±1.77	89.89±0.94	90.35±1.25	90.72±0.63	95.69±0.47	95.17±0.57	95.36±0.51
	subgraph	92.18±1.34	91.89±2.03	91.85±1.60	84.49±1.55	85.01±1.03	84.99±1.78	90.14±0.97	91.05±1.26	90.49±0.97	95.88±0.86	95.58±0.96	95.74±0.63
SAMamba	subtree	92.53±1.51	91.03±1.09	91.88±1.12	84.67±1.23	84.88±1.23	84.03±1.51	90.20±0.77	90.45±0.93	90.66±0.44	95.90±0.36	95.64±1.00	96.01±0.45
	subgraph	**93.25±1.71**	**92.29±2.32**	**92.97±1.17**	**86.16±0.96**	**86.34±1.37**	**85.12±1.92**	**91.97±1.61**	**92.75±1.04**	**91.56±0.37**	**97.44±0.15**	**97.09±0.13**	**97.45±0.21**

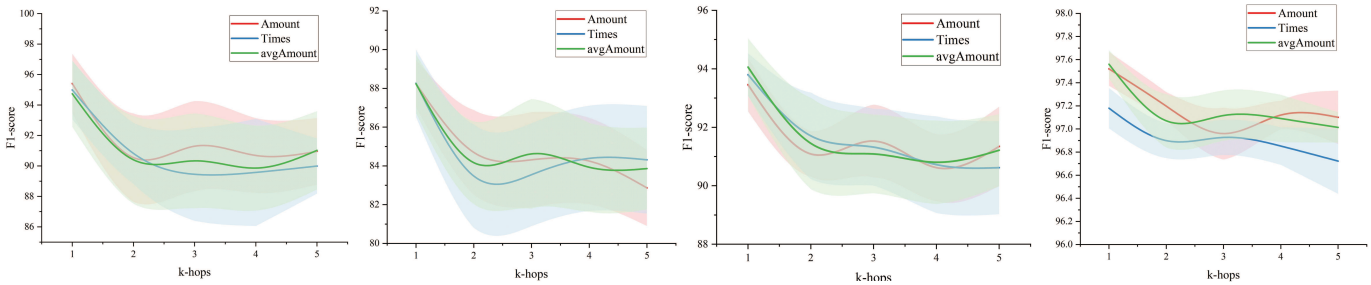
TABLE VI

FLOPS, PARAMS AND MEMORY BENCHMARKS ON THE ETH-PHISH&HACK (AMOUNT) SUB-DATASET

Method	FLOPs (GB)↓	Params (MB)↓	Memory (MB)↓
I2BGNNA	0.00084851	0.97	41.28
SGFormer	0.18734515	1.94	59.35
GPSPerformer	0.10051632	1.04	41.89
GPSTransformer	0.09733782	1.04	41.69
Ethident	0.09521440	1.00	40.86
FAGNN	0.18603315	1.92	58.57
AEtransGAT	0.18666176	1.91	58.31
SAMamba	0.09738912	1.09	42.57

LSP module adopts subgraph encoding or subtree encoding. The LSP module adopts subgraph encoding and integrates the GIS module to obtain overall performance improvement. This finding suggests two insights: on the one hand, the subgraph encoding strategy of the LSP module effectively captures the behavior patterns of accounts, but lacks the discrimination of different behavior patterns. On the other hand, the GIS module is proficient in discriminating the differences of all behavior patterns through global and extensive analysis.

2) Evaluating the Effectiveness of LSP Module: We conduct ablation experiments to study the impact of the LSP module on account classification performance, and Table V reports the results in terms of Micro-F1 score. The SAMamba framework that introduces the LSP module achieves an overall performance improvement. This is because the GIS module fails to learn from the graph topology and simply learns node features as its classification basis. However, topological features are crucial information for describing account behavior characteristics and provide important basis for accurate account classification. In contrast, the LSP module learns



(a) Eth-ICO (b) Eth-Mining (c) Eth-Exchange (d) Eth-Phish&Hack

Fig. 5. Performance of SAMamba with different k-hops hyperparameters on the lw-AIG dataset. Three-fold cross-validation was conducted, and the average Micro F1-score was reported with the filled area representing the standard deviation.

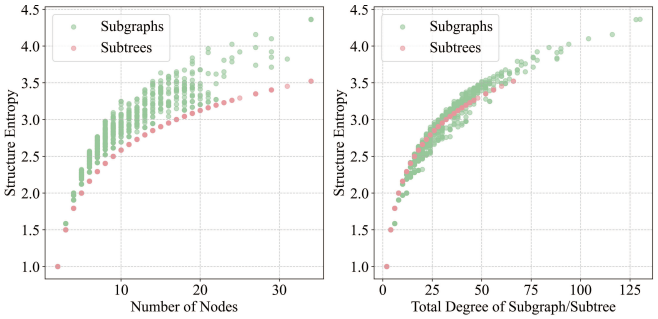
account behavior patterns from the graph topology and cooperates with the GIS module to discriminate the relationship between specific behaviors and account types.

3) Evaluating the Effectiveness of GIS Module: In order to explore the contribution of the GIS module in improving account classification performance, we conduct ablation experiments of the GIS module and report the experimental results in the form of F1-score in Table V. The results show that the integration of GIS modules in the SAMamba framework leads to significant performance improvements. This effect results from the GIS module modeling long-distance dependencies in heterogeneous graphs and adaptively filtering appropriate information through a selection mechanism. In addition, the effective coupling of the LSP module and the GIS module enables the simultaneous capture of similarities in transaction patterns and features, empowering SAMamba with excellent account classification capabilities.

4) Evaluating the Effectiveness of Local Attention Mechanism: Table V provides ablation results on how local attention affects the performance of account classification, reporting the F1 scores obtained by LSP and LSP without local attention (w/o LA). LSP does not achieve significant positive gains compared to LSP (w/o LA). We further study how local attention affects the performance of the SAMamba framework and empirically observe that the integration of local attention improves the performance of SAMamba. This result shows that focusing only on feature correlations in the local range is insufficient, and adaptive routing combined with relevant information in the global range leads to optimal performance.

H. Hyperparameter Studies

Consider a large-scale Ethereum account interaction graph, where the target node must pay attention to nodes of the same class that are k-hops away. A naive solution is to stack multiple layers of LSP, repeatedly aggregating information from 1-hop neighbors. However, this solution indiscriminately aggregates signals from nodes of different classes that are k-hops away, causing performance to degrade dramatically [52]. To investigate how LSP and GIS play their respective advantages, we report the performance of LSP in terms of F1-score under different k-hops settings. The experimental results are shown in the Fig. 5, where the filled areas represent the standard deviation. It is worth noting that the performance



(a) Structure Entropy vs. Substructure Size (b) Structure Entropy vs. Total Degree

Fig. 6. Comparative analysis of structural entropy between subgraph and subtree structures, derived from statistical analysis of a batch of data from the Eth-Phish&Hack subdataset. (a) Comparison of structural entropy between subgraphs and subtrees under different sizes. (b) Comparison of structural entropy between subgraphs and subtrees under different total degrees.

gradually decreases with the increase of LSP depth, which is consistent with intuition. Multi-layer stacking of LSP causes node features to become equivalent on the entire graph, i.e., over-smoothing, thereby weakening the information selection capability of GIS. In contrast, LSP with a smaller k-hops hyperparameter settings preserves the structure, allowing GIS to filter out unimportant information while ensuring the distance of information propagation. This synergy balances the complex structure perception of LSP and the information adaptive selection of GIS, achieving better long-range learning.

I. Structural Entropy Analysis

To intuitively demonstrate the design philosophy of the node prioritization strategy and validate its rationality, we introduce degree-based structural entropy [53] to quantitatively compare the information differences between subgraph and subtree structures under different sizes and total degrees. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a given subgraph or subtree, and its structural entropy is defined as follows:

$$I(\mathcal{G}, k) = -\sum_{i=1}^{|\mathcal{V}|} \frac{d_i^k}{\sum_{j=1}^{|\mathcal{V}|} d_j^k} \log\left(\frac{d_i^k}{\sum_{j=1}^{|\mathcal{V}|} d_j^k}\right) = \log\left(\sum_{i=1}^{|\mathcal{V}|} d_i^k\right) - \sum_{i=1}^{|\mathcal{V}|} \frac{d_i^k}{\sum_{j=1}^{|\mathcal{V}|} d_j^k} \log d_i^k \quad (20)$$

(a) Eth-ICO (b) Eth-Mining (c) Eth-Exchange (d) Eth-Phish&Hack

Fig. 7. t-SNE plot of the graph embedding generated by SAMamba on the lw-AIG dataset. It shows excellent separability in different sub-datasets.

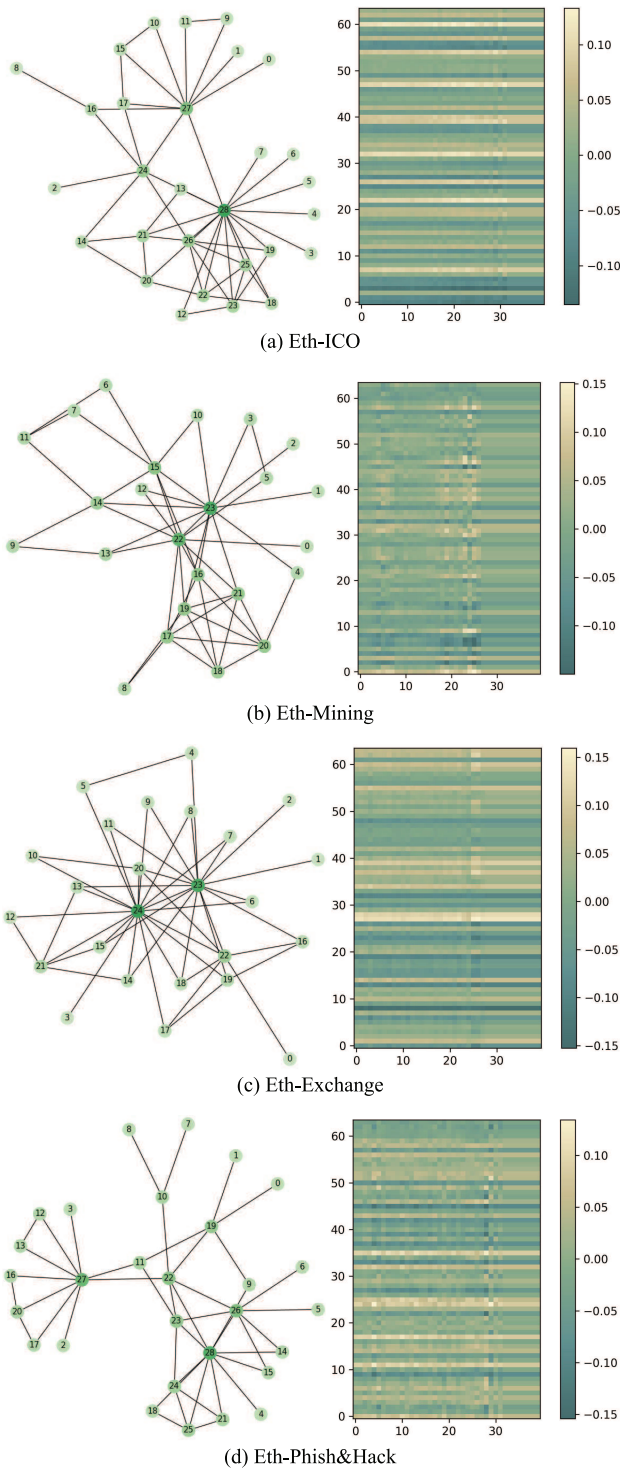
where d_i is the degree of node $v_i \in \mathcal{V}$, k represents the power parameter, which is set to 1. Based on the aforementioned definition, a higher structural entropy signifies a more uniform node degree distribution, minimal disparity in the importance of nodes, and a structural manifestation of disorder. Conversely, a lower structural entropy denotes conspicuous variations in node degree distribution, where a minority of high-degree hub nodes dominate the network topology, exhibiting structural order. As illustrated in Fig. 6(a), the structural entropy of subgraphs significantly exceeds that of subtrees across various sizes, which substantiates that the core value of subgraph structures lies in their ability to transcend the limited expressiveness of subtree structures. We further analyze the differences in structural entropy between subgraphs and subtrees under different total degrees. As shown in Fig. 6(b), structural entropy exhibits an upward trend with the increase of degrees, which justifies the rationality of the node priority strategy. As a direct quantitative indicator of node connections in subgraphs, degree can capture the density of local neighborhoods and reflect the role of connectivity hubs in global networks. Therefore, the sorting strategy based on subgraph degrees essentially achieves more accurate node importance output through more complete structural information input. Another insight derived from Fig. 6(b) is that subgraphs can accumulate larger total degrees compared to subtree structures, thereby obtaining higher structural entropy to mine more subtle structural information.

J. Visualization

This section visualizes the learned graph embeddings and key parameters to provide intuitive insight into SAMamba's ability to interpret graph properties and structures.

1) t-SNE Visualization of Account Representation: We employ a widely used data visualization method, t-distributed Stochastic Neighbor Embedding (t-SNE) [54], to display the embeddings learned by the SAMamba. t-SNE projects high-dimensional data onto a two-dimensional plane, showcasing how our SAMamba framework clusters data points. As shown in Fig. 7, experiments are conducted on Eth-ICO, ETH-Mining, ETH-Exchange, and ETH-Phish&Hack, with a focus on the separability of the embeddings learned for different strategy of samples. The t-SNE results in Fig. 7 also validate that the graph embeddings learned by SAMamba are separable, confirming its superior performance.

2) Visualization of Patterns and Selection Mechanisms: Mechanistically, Δ controls the balance between the current input x_t and the previous state h_{t-1}. A large Δ focuses on the current input x_t to update the previous state h_{t-1}, while a small Δ ignores the current input x_t and persists the previous state h_{t-1}. In order to observe how the selection mechanism works to control complex structural patterns, we extract pattern examples from Eth-ICO, Eth-Mining, Eth-Exchange, Eth-phish&Hack and report Δ on these graphs. As shown in Fig. 8, the Ethereum account transaction graph consists of many possible substructures, including cycle structures. The structure-awareness ability of LSP allows SAMamba to learn structural representations of different specific substructures, thereby achieving accurate classification of these differentiated structural representations. Fig. 8 is consistent with our conjecture, where the part corresponding to the high-degree nodes in these Δ matrices is observed to oscillate. Another insight from this result is that high-degree nodes aggregate differentiated features from different classes of neighboring nodes, thereby facilitating GIS to regulate



(a) Eth-ICO

(b) Eth-Mining

(c) Eth-Exchange

(d) Eth-Phish&Hack

Fig. 8. Example graph and Δ matrices retrieved from SAMamba. The vertical axis corresponds to the node number, and the horizontal axis corresponds to the neighbor node number. The vertical axis corresponds to the dimension of Δ.

Δ to selectively focus on or ignore information on hidden states.

V. Conclusion and Discussion

This paper studies the fraud detection problem based on the behavior patterns of Ethereum users and proposes the structure-aware fraud detection framework SAMamba. Compared with existing graph neural network-based methods, SAMamba shows obvious advantages in modeling local behavior patterns and global interactions. In SAMamba, we design a local structure preservation module (i.e., LSP) and a global information selection module (i.e., GIS). Specifically, LSP introduces a novel local attention-based subgraph encoding strategy for capturing potential transaction links. As a powerful global reasoning module, GIS selectively routes task-related information in heterophilic Ethereum networks, thereby improving the performance of downstream tasks. Extensive experimental results on EPTransNet and lw-AIG datasets demonstrate the effectiveness of SAMamba. Although these achievements, SAMamba is still confronted with certain limitations, which impel us to engage in continuous iterative optimization in future research to address the ever-evolving complexity of fraudulent activities. Specifically, the current sophisticated fraudsters persistently update their criminal tactics to evade detection, presenting a challenge whereby SAMamba, despite having captured diverse complex behavioral patterns, remains insufficient in dealing with hitherto unseen novel criminal approaches. In future work, the exploration of how to leverage incremental learning techniques to progressively update knowledge about new criminal patterns without compromising SAMamba's pre-existing knowledge base warrants further investigation.

References

[1] N. Ivanov, Q. Yan, and A. Kompalli, "TxT: Real-time transaction encapsulation for Ethereum smart contracts," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 1141–1155, 2023.

[2] H. Tian et al., "Enabling cross-chain transactions: A decentralized cryptocurrency exchange protocol," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 3928–3941, 2021.

[3] L. Chen, J. Peng, Y. Liu, J. Li, F. Xie, and Z. Zheng, "Phishing scams detection in Ethereum transaction network," *ACM Trans. Internet Technol.*, vol. 21, no. 1, pp. 1–16, Dec. 2020.

[4] T.-H. Chang and D. Svetinovic, "Improving Bitcoin ownership identification using transaction patterns analysis," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 1, pp. 9–20, Jan. 2020.

[5] S. Farrugia, J. Ellul, and G. Azzopardi, "Detection of illicit accounts over the Ethereum blockchain," *Expert Syst. Appl.*, vol. 150, Jul. 2020, Art. no. 113318.

[6] F. Victor, "Address clustering heuristics for Ethereum," in *Proc. Financ. Cryptogr. Data Secur.* Cham, Switzerland: Springer, Feb. 2020, pp. 617–633.

[7] S. Linoy, N. Stakhanova, and S. Ray, "De-anonymizing Ethereum blockchain smart contracts through code attribution," *Int. J. Netw. Manage.*, vol. 31, no. 1, p. 2130, Jan. 2021.

[8] R. Agarwal, T. Thapliyal, and S. K. Shukla, "Detecting malicious accounts showing adversarial behavior in permissionless blockchains," 2021, *arXiv:2101.11915*.

[9] C. Wang et al., "Demystifying Ethereum account diversity: Observations, models and analysis," *Frontiers Comput. Sci.*, vol. 16, no. 4, pp. 1–12, Aug. 2022.

[10] Y. Pang et al., "Graph decipher: A transparent dual-attention graph neural network to understand the message-passing mechanism for the node classification," *Int. J. Intell. Syst.*, vol. 37, no. 11, pp. 8747–8769, Jul. 2022.

[11] S. Li, G. Gou, C. Liu, C. Hou, Z. Li, and G. Xiong, "TTAGN: Temporal transaction aggregation graph network for Ethereum phishing scams detection," in *Proc. ACM Web Conf.*, Apr. 2022, pp. 661–669.

[12] H. Zheng, M. Ma, H. Ma, J. Chen, H. Xiong, and Z. Yang, "TEGDetector: A phishing detector that knows evolving transaction behaviors," *IEEE Trans. Computat. Social Syst.*, vol. 11, no. 3, pp. 3988–4000, Jun. 2024.

[13] D. Lin, J. Wu, T. Huang, K. Lin, and Z. Zheng, "Who is who on Ethereum? Account labeling using heterophilic graph convolutional network," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 54, no. 3, pp. 1541–1553, Mar. 2024.
[14] J. Liu, J. Chen, J. Wu, Z. Wu, J. Fang, and Z. Zheng, "Fishing for fraudsters: Uncovering Ethereum phishing gangs with blockchain data," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 3038–3050, 2024.
[15] S. Hu, Z. Zhang, B. Luo, S. Lu, B. He, and L. Liu, "BERT4ETH: A pre-trained transformer for Ethereum fraud detection," in *Proc. ACM Web Conf.*, Apr. 2023, pp. 2189–2197.
[16] D. Zeng, W. Liu, W. Chen, L. Zhou, M. Zhang, and H. Qu, "Substructure aware graph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, Jun. 2023, vol. 37, no. 9, pp. 11129–11137.
[17] M. Zhang and P. Li, "Nested graph neural networks," in *Proc. Adv. Neural. Inf. Process. Syst.*, Jan. 2021, pp. 15734–15747.
[18] J. Shen, J. Zhou, Y. Xie, S. Yu, and Q. Xuan, "Identity inference on blockchain using graph neural network," in *Proc. Int. Conf. Blockchain Trustworthy Syst.*, in Communications in Computer and Information Science, Aug. 2021, pp. 3–17.
[19] J. Zhou, C. Hu, J. Chi, J. Wu, M. Shen, and Q. Xuan, "Behavior-aware account de-anonymization on Ethereum interaction graph," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 3433–3448, 2022.
[20] J. Liu, J. Zheng, J. Wu, and Z. Zheng, "FA-GNN: Filter and augment graph neural networks for account classification in Ethereum," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 4, pp. 2579–2588, Jul. 2022.
[21] L. Yu, F. Zhang, J. Ma, L. Yang, Y. Yang, and W. Jia, "Who are the money launderers? Money laundering detection on blockchain via mutual learning-based graph neural network," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jun. 2023, pp. 1–8.
[22] J. Feng, Y. Chen, F. Li, A. Sarkar, and M. Zhang, "How powerful are K-hop message passing graph neural networks," in *Proc. Adv. Neural. Inf. Process. Syst.*, Jan. 2022, pp. 4776–4790.
[23] Z. Wu, P. Jain, M. A. Wright, A. Mirhoseini, J. E. Gonzalez, and I. Stoica, "Representing long-range context for graph neural networks with global attention," in *Proc. Adv. Neural. Inf. Process. Syst.*, Jan. 2022, pp. 13266–13279.
[24] T. Huang, D. Lin, and J. Wu, "Ethereum account classification based on graph convolutional network," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 5, pp. 2528–2532, May 2022.
[25] A. Behrouz and F. Hashemi, "Graph mamba: Towards learning on graphs with state space models," 2024, *arXiv:2402.08678*.
[26] J. Wu et al., "Who are the phishers? Phishing scam detection on Ethereum via network embedding," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 2, pp. 1156–1166, Feb. 2022.
[27] 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.
[28] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 855–864.
[29] D. Lin, J. Wu, Q. Yuan, and Z. Zheng, "Modeling and understanding Ethereum transaction records via a complex network approach," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 67, no. 11, pp. 2737–2741, Nov. 2020.
[30] D. Lin, J. Wu, Q. Yuan, and Z. Zheng, "T-EDGE: Temporal WEighted MultiDiGraph embedding for Ethereum transaction network analysis," *Frontiers Phys.*, vol. 8, p. 204, Jun. 2020.
[31] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.
[32] P. Velič ković, G. Cucurull, A. Casanova, A. Romero, P. Lió, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*.
[33] B. Bevilacqua et al., "Equivariant subgraph aggregation networks," *arXiv:2110.02910*.
[34] L. Zhao, W. Jin, L. Akoglu, and N. Shah, "From stars to subgraphs: Uplifting any GNN with local structure awareness," *arXiv:2110.03753*.
[35] F. Frasca, B. Bevilacqua, M. M. Bronstein, and H. Maron, "Understanding and extending subgraph GNNs by rethinking their symmetries," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2022, pp. 31376–31390.
[36] G. Xue, M. Zhong, T. Qian, and J. Li, "PSA-GNN: An augmented GNN framework with priori subgraph knowledge," *Neural Netw.*, vol. 173, May 2024, Art. no. 106155.
[37] A. Leman and B. Weisfeiler, "A reduction of a graph to a canonical form and an algebra arising during this reduction," *Nauchno-Technicheskaya Informatsiya*, vol. 2, no. 9, pp. 12–16, 1968.
[38] T. Konstantin Rusch, M. M. Bronstein, and S. Mishra, "A survey on oversmoothing in graph neural networks," 2023, *arXiv:2303.10993*.
[39] J. Topping, F. Di Giovanni, B. Paul Chamberlain, X. Dong, and M. M. Bronstein, "Understanding over-squashing and bottlenecks on graphs via curvature," 2021, *arXiv:2111.14522*.
[40] F. D. Giovanni, L. Giusti, F. Barbero, G. Luise, P. Lió, and M. M. Bronstein, "On over-squashing in message passing neural networks: The impact of width, depth, and topology," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Jan. 2023, pp. 7865–7885.
[41] U. Alon and E. Yahav, "On the bottleneck of graph neural networks and its practical implications," 2020, *arXiv:2006.05205*.
[42] V. P. Dwivedi et al., "Long range graph benchmark," in *Proc. Adv. Neural. Inf. Process. Syst.*, Jan. 2022, pp. 22326–22340.
[43] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph transformer networks," in *Proc. Adv. Neural. Inf. Process. Syst.*, vol. 32, Nov. 2019, pp. 11960–11970.
[44] V. Prakash Dwivedi and X. Bresson, "A generalization of transformer networks to graphs," 2020, *arXiv:2012.09699*.
[45] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini, "Recipe for a general, powerful, scalable graph transformer," in *Proc. Adv. Neural. Inf. Process. Syst.*, Jan. 2022, pp. 14501–14515.
[46] F. Shi et al., "Large language models can be easily distracted by irrelevant context," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Jan. 2023, pp. 31210–31227.
[47] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *Proc. NIPS*, vol. 33, 2020, pp. 5812–5823.
[48] P. Zheng, Z. Zheng, J. Wu, and H.-N. Dai, "XBlock-ETH: Extracting and exploring blockchain data from Ethereum," *IEEE Open J. Comput. Soc.*, vol. 1, pp. 95–106, 2020.
[49] Q. Wu et al., "SGFormer: Simplifying and empowering transformers for large-graph representations," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2023, pp. 64753–64773.
[50] A. Vaswani et al., "Attention is all you need," *Adv. Neural Inf. Process. Syst.*, vol. 30, pp. 5998–6008, Jun. 2017.
[51] K. Choromanski et al., "Rethinking attention with performers," 2020, *arXiv:2009.14794*.
[52] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018.
[53] S. Cao, M. Dehmer, and Y. Shi, "Extremality of degree-based graph entropies," *Inf. Sci.*, vol. 278, pp. 22–33, Sep. 2014.
[54] L. V. D. Maaten and G. E. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 86, pp. 2579–2605, Jan. 2008.