Profit or Deceit? Mitigating Pump and Dump in DeFi via Graph and Contrastive Learning

Cong Wu, Jing Chen, Jiahong Li, Jiahua Xu, Ju Jia, Yutao Hu, Yebo Feng, Yang Liu, and Yang Xiang

***Abstract*—Pump-and-Dump (PD) schemes pose a significant threat to the stability and fairness of Decentralized Finance (DeFi) markets, often resulting in substantial financial losses for investors. The early and accurate detection of these schemes is crucial for preserving trust in the rapidly expanding cryptocurrency ecosystem. However, existing detection methods primarily rely on post-event analysis and heuristic-based approaches, which are often inadequate for real-time and precise identification of PD activities. In this paper, we present PUMPWATCHER, an innovative framework that employs Graph Neural Networks (GNNs) and contrastive learning to detect PD schemes by modeling transaction behaviors within temporal graphs. PUMPWATCHER integrates advanced transaction graph construction, temporal GNNs, and contrastive learning techniques to enhance node and edge representations, thereby improving the detection of intricate and covert PD operations. We validate PUMPWATCHER on a dataset from Uniswap, encompassing 924,508 transactions across 858 tokens within December 2022. The results show that PUMPWATCHER outperforms state-of-the-art models, achieving a superior balanced accuracy of 92.3%, while significantly minimizing false positives and negatives. These outcomes highlight its potential to set a new standard in real-time detection of market manipulation, paving the way for more secure and resilient DeFi ecosystems.**

***Index Terms*—Decentralized finance, pump-and-dump, graph neural networks, contrastive learning.**

Received 28 August 2024; revised 17 May 2025 and 22 June 2025; accepted 21 July 2025. Date of publication 1 August 2025; date of current version 29 August 2025. This work was supported in part by the National Research Foundation, Singapore, and the Cyber Security Agency under its National Cybersecurity Research and Development Program under Grant NCRP25-P04-TAICeN; in part by the Defence Science Organisation (DSO) National Laboratories under the AI Singapore Program (AISG) under Award AISG2-GC-2023-008; in part by the National Research Foundation, Prime Minister's Office, Singapore, through the Campus for Research Excellence and Technological Enterprise (CREATE) Program; in part by the National Natural Science Foundation of China under Grant 62402106; in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK20241272; in part by the National Key Research and Development Program of China under Grant 2021YFB2700200; and in part by UK Centre for Blockchain Technologies through Ripple's University Blockchain Research Initiative (UBRI). The associate editor coordinating the review of this article and approving it for publication was Prof. Kwok-Yan Lam. *(Corresponding authors: Ju Jia; Yutao Hu.)*

Cong Wu, Jing Chen, and Jiahong Li are with the Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China (e-mail: cnacwu@whu.edu.cn; chenjing@whu.edu.cn; furina@whu.edu.cn).

Jiahua Xu is with the Centre for Blockchain Technologies, University College London, WC1E 6BT London, U.K., and also with the DLT Science Foundation, WC2H 2JQ London, U.K. (e-mail: jiahua.xu@ucl.ac.uk).

Ju Jia is with the School of Cyber Science and Engineering, Southeast University, Nanjing 210096, China (e-mail: jiaju@seu.edu.cn).

Yutao Hu is with the School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: yutaohu@hust.edu.cn).

Yebo Feng and Yang Liu are with the College of Computing and Data Science, Nanyang Technological University, Singapore 639798 (e-mail: yebo.feng@ntu.edu.sg; yangliu@ntu.edu.sg).

Yang Xiang is with the Digital Research, Swinburne University of Technology, Melbourne, VIC 3122, Australia (e-mail: yxiang@swin.edu.au).

Digital Object Identifier 10.1109/TIFS.2025.3594873

I. INTRODUCTION

BLOCKCHAIN and DeFi has revolutionized the financial industry by enabling peer-to-peer transactions without the need for traditional intermediaries [1], [2], [3], [4], [5]. As of May 2025, the total value locked (TVL) in DeFi reached $ 250 billion [6], underscoring its significant impact. However, the rapid growth and inherent anonymity of DeFi have also made it a target for various forms of market manipulation [7], [8]. Among these, pump-and-dump (PD) schemes [9], [10], [11] are particularly concerning, as they involve artificially inflating the price of a cryptocurrency, often ERC-20 tokens [12], the primary form of DeFi crypto assets, through coordinated buying, followed by a sudden sell-off to profit from the manipulated prices [13]. These schemes not only undermine the integrity of DeFi markets but also lead to significant financial losses for unsuspecting investors, making the detection and prevention of PD activities a critical area of research.

Existing research on PD schemes in DeFi has concentrated on detecting abnormal trading behaviors and market manipulation through various analytical approaches [9], [10], [11], [14], [15], [16]. Initial studies utilized statistical models and heuristic-based methods to identify price anomalies by analyzing historical trading data and transaction patterns [9], [14]. These methods, while effective in post-event analysis, are limited by their reliance on pre-defined thresholds and rules, which restricts their applicability for real-time detection. To enhance detection accuracy, transnational features, e.g., trading volumes, price movements, and social media activities are incorporated with traditional machine learning classifiers [11], [16]. However, these models often face scalability challenges, particularly when applied across a wide array of ERC-20 tokens with varying liquidity and trading behaviors. Besides, advanced deep learning-based approaches, including convolutional and recurrent neural networks, have been explored to capture temporal dynamics and complex dependencies in trading data [15]. Despite their potential, these deep learning models require extensive computational resources and large datasets, limiting their practical use in rapidly evolving DeFi markets.

A. Motivation

Despite growing research on detecting PD schemes, existing methods suffer from key practical limitations. They often rely on static thresholds or hand-crafted heuristics, which struggle to adapt to the dynamic, fast-moving nature of DeFi

markets [17]. Additionally, their dependency on large volumes of labeled training data restricts scalability across the diverse and ever-changing ERC-20 token landscape. Many are also computationally intensive, limiting their applicability in real-time scenarios. In contrast, our method combines graph learning [18] and contrastive learning (CL) [19], [20] to model evolving token behaviors with minimal supervision and high generalization.

B. Our Approach

In this paper, we propose PUMPWATCHER, the first temporal graph learning-based system for detecting PD in DeFi markets. The key idea behind PUMPWATCHER is to leverage temporal GNNs to characterize the evolving, graph-structured behavior of token transactions, capturing both the structural and temporal dependencies that are crucial for identifying manipulative activities. To further enhance the detection accuracy, it integrates CL, which not only refines the learned representations but also reduces the dependence on large labeled training datasets. The framework operates by first constructing detailed transaction graphs for each ERC-20 token, which encapsulate the trading activities and interactions over time. These graphs are then processed through the PD-GNN module, where temporal dynamics are learned, followed by CL to optimize the feature representations. Finally, the refined representations are fed into a LightGBM classifier, which accurately distinguishes between normal and PD-related transactions, providing a robust solution for real-time PD detection in DeFi ecosystems.

C. Practical Deployment

In real-world scenarios, PUMPWATCHER can be effectively implemented as a real-time monitoring tool in DeFi platforms, particularly on exchanges like Uniswap. It continuously captures and processes transaction data, constructing dynamic transaction graphs for each ERC-20 token. These graphs are analyzed by the PD-GNN model to detect emerging PD schemes. By incorporating CL, PUMPWATCHER operates efficiently with limited labeled data, making it adaptable to various market conditions. Deploying PUMPWATCHER directly on blockchain nodes or within exchange infrastructure enables timely and accurate detection of PD activities, helping maintain market integrity and protect investors. In real-testing, PUMPWATCHER was deployed in MetaTrust to detect PD schemes on Ethereum from August 1 to 20, 2024. It successfully identified 7,562 PD transactions across 181 ERC-20 tokens. Additionally, PUMPWATCHER flagged 4,198 transactions as suspicious, pending further investigation, with initial validations by blockchain auditors indicating a high probability of PD activity. Notably, PUMPWATCHER contributed to preventing potential investor losses estimated at $ 75,200. All detected PD schemes were responsibly disclosed to Web3 security communities, including MetaTrust and ChainUp.

D. Novelty Over Existing Works

Compared to existing PD detection methods, PUMPWATCHER introduces several key innovations to achieve robust and accurate detection in open-world settings. (i) PUMPWATCHER is the first framework to leverage temporal GNN for PD detection in DeFi markets, effectively modeling the dynamic and complex transaction behaviors over time. (ii) It employs novel temporal graph learning techniques that capture both the structural and temporal dependencies within transaction data, enabling more precise and timely identification of manipulative activities. (iii) PUMPWATCHER integrates CL to refine the learned representations, enhancing the distinction between normal and fraudulent transactions while reducing the reliance on large-scale labeled datasets, thereby improving scalability and adaptability across diverse ERC-20 tokens.

E. Contributions

We make the following contributions:

- We propose PUMPWATCHER, the first temporal GNN-based system for detecting PD schemes in DeFi markets. This approach effectively captures the evolving and complex behaviors of token transactions over time.
- We develop PD-GNN, a temporal GNN model within PUMPWATCHER that integrates structural and temporal transaction features. By incorporating CL, it enhances representation quality, improving PD detection accuracy and reducing reliance on extensive labeled data.
- We compile a large publicly available dataset containing 924,508 transactions across 858 different tokens, with approximately 13% of transactions labeled as PD.
- We conduct comprehensive evaluations of PUMPWATCHER using real-world DeFi transaction data and comparison with existing methods. Results demonstrates that PUMPWATCHER outperforms state-of-the-art PD detections, significantly reducing false positives and negatives. We also real-test PUMPWATCHER on Ethereum and present case studies of real PD.

F. Code Availability

We aim to promote open, reproducible, and transparent research among the academic research community. We publish the dataset and codebase of PUMPWATCHER, which can be obtained at GitHub repository: https://github.com/Confringo233/PumpWatcher.

II. BACKGROUND

In this section, we present background of PD and GNN.

A. Pump and Dump in DeFi

PD schemes in this work refer specifically to coordinated manipulative behaviors where the price of a token is artificially inflated through synchronized buying activity and hype generation, followed by a rapid sell-off for profit [8], [11]. While speculative trading and volatility are inherent features of both traditional and decentralized markets, we focus on cases where market participants intentionally deceive others, often by spreading misleading information or executing trades across multiple pseudonymous accounts, to create a false

sense of demand [16], [21]. These manipulative behaviors are particularly harmful in DeFi due to the absence of centralized oversight, rapid transaction speeds, and low liquidity, which amplify the impact of even small amounts of capital. Unlike in traditional markets, where such activities may be mitigated by regulation or market mechanisms, DeFi markets are vulnerable to PD schemes, leading to significant market disruption, investor losses, and long-term systemic risks. These events typically unfold in three stages: setup (accumulation), pump (hype-driven surge), and dump (rapid liquidation), often resulting in sharp price collapses and substantial losses for late participants. By targeting this class of manipulations, our work aims to improve transparency, reduce systemic risk, and support fairer participation in decentralized financial systems.

It is important to distinguish PD behavior in DeFi from similar speculative patterns observed in traditional financial markets [8], [21]. In regulated environments such as equity markets, mechanisms like circuit breakers, centralized surveillance, and legal consequences help contain the systemic impact of short-term manipulative activity. Moreover, PD-like behaviors in small-cap or penny stocks often occur within a tightly monitored ecosystem, allowing for post-event regulatory action. In contrast, DeFi ecosystems are open, permissionless, and largely pseudonymous, lacking centralized oversight or enforcement. This creates a unique vulnerability where coordinated manipulation can be executed at scale, often across multiple tokens and platforms, with minimal risk of accountability. As a result, what might be considered a fringe market anomaly in traditional finance can escalate into a systemic risk in DeFi, undermining market integrity, investor trust, and token utility.

The detection and prevention of PD schemes in cryptocurrency markets have emerged as critical research areas. Unlike traditional financial markets, which are regulated by entities capable of monitoring and mitigating such activities, the decentralized and pseudonymous nature of cryptocurrency trading introduces unique challenges. Researchers have adopted various techniques to address these challenges, including machine learning models that analyze trading patterns, social media signals, and other market indicators. These approaches are designed to identify anomalous trading behaviors and market conditions that signal the presence of a PD scheme. The integration of diverse data sources and advanced algorithms has demonstrated considerable potential in enhancing the accuracy and timeliness of PD detection, thereby offering better protection for investors and contributing to the overall integrity of cryptocurrency markets.

B. Graph Neural Networks

GNNs are a class of neural networks tailored for graph-structured data, which are prevalent in domains such as social networks, knowledge graphs, traffic systems, and molecular structures [22], [23]. Among GNN architectures, convolutional GNNs are particularly noteworthy for their neighborhood aggregation strategy, often referred to as message passing. This approach iteratively refines node representations by integrating a node's feature vector with those of its neighbors, similar to convolution operations in Convolutional Neural Networks (CNNs). This strategy enables GNNs to capture complex dependencies within graph data, making them effective in learning rich node and edge representations.

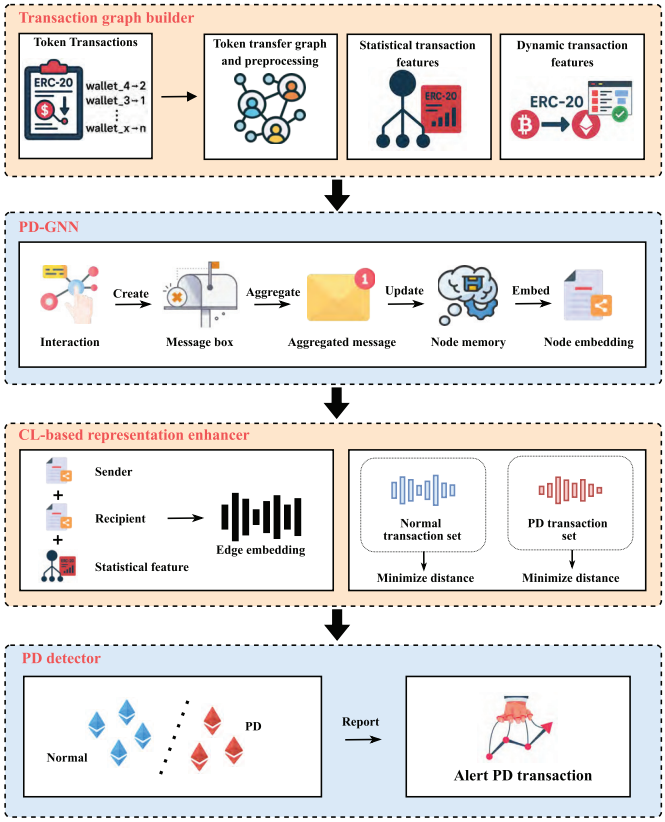


Fig. 1. Overview of PUMPWATCHER.

Formally, given a graph $G = (V, E)$ with nodes $v \in V$ and edges $e \in E$, where each node v has a feature vector $\mathbf{x}_v$, the GNN updates the node representation $\mathbf{h}_v^{(k)}$ at layer k as:

$$\mathbf{h}_v^{(k+1)} = \sigma\left(\mathbf{W}^{(k)}\mathbf{h}_v^{(k)} + \sum_{u \in \mathcal{N}(v)} \mathbf{W}_{\text{neigh}}^{(k)}\mathbf{h}_u^{(k)}\right),$$

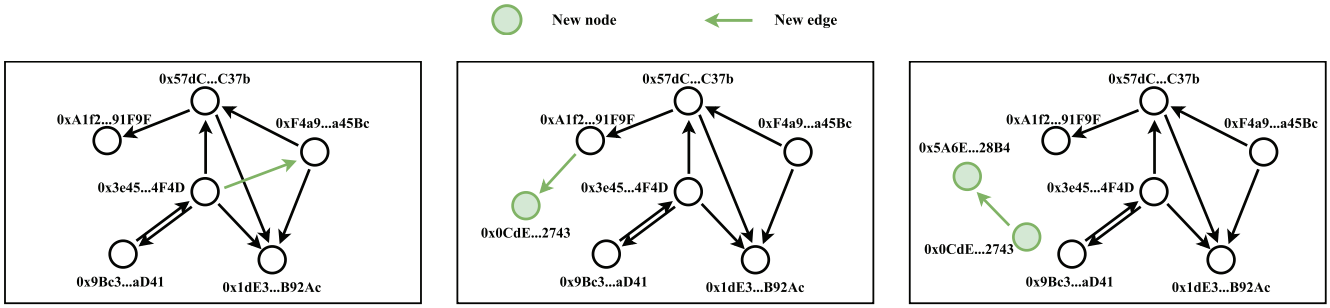
where $\mathbf{W}^{(k)}$ and $\mathbf{W}_{\text{neigh}}^{(k)}$ are learnable weight matrices, $\mathcal{N}(v)$ denotes the neighbors of v, and σ is a non-linear activation function. Initially, $\mathbf{h}_v^{(0)} = \mathbf{x}_v$. GNNs leveraging this approach have achieved state-of-the-art performance across domains such as point cloud classification [24], recommendation systems [25], spam detection [26], [27], and molecular applications [28], [29], making them particularly suitable for tasks like detecting PD schemes in cryptocurrency markets.

III. PUMPWATCHER

This section presents overview and detail each module.

A. Overview

As illustrated in Figure 1, PUMPWATCHER consists of four key modules: the Transaction graph builder, PD-GNN, CL-based representation enhancer, and PD detector. The Transaction graph builder constructs individual transaction



(a) Shared source and target nodes (b) Only one shared node (c) No shared nodes

Fig. 2. Illustration of transaction graph construction.

graphs for each ERC-20 token, capturing unique trading characteristics. The PD-GNN processes these graphs by integrating temporal information to model the evolving dynamics of transactions, enabling effective node representation learning. The CL-based representation enhancer refines these representations through contrastive learning, improving the model's ability to distinguish between normal and PD-related activities. Finally, the PD detector classifies transactions using LightGBM, a supervised learning algorithm trained on labeled data indicating PD-related transactions, chosen for its efficiency and accuracy in handling large-scale data, leading to precise detection and mitigation of PD schemes in DeFi markets.

B. Transaction Graph Builder

This module transforms ERC-20 token transfer records into temporal transaction graphs to capture the evolving trading behavior of each token. Specifically, we construct one directed graph per token, where each node represents a unique wallet address that has participated in transactions involving the token. Directed edges represent token transfers, with each edge annotated by a timestamp, transaction value, and other contextual features. The temporal nature of the graph allows us to model trading behaviors over time, enabling the GNN to learn both structural and dynamic transaction patterns.

To extract features from the incoming transaction data, we first identify the involved account nodes. Then, based on the node-sharing relationship between the new transaction and the existing transaction graph, we determine whether new nodes need to be added. Subsequently, a new edge is added and annotated with relevant transaction attributes (including transaction amount, time, token type, etc.). As illustrated in Figure 2, the graph construction and update process involves three scenarios:

- *Both source and destination nodes are shared*: a new edge is added between the existing nodes.
- *Only one node is shared*: a new node is added, and an edge is created between the new node and the existing one.
- *No node is shared*: two new nodes are introduced, and an edge is established between them.

Currently, edges are limited to direct token transfers i.e., transactions in which tokens move explicitly from one address to another. We do not incorporate indirect relationships or inferred links, such as intermediary wallets that may participate in multi-hop paths between source and destination accounts. While this design simplifies the graph structure and improves computational tractability, we acknowledge that modeling indirect or latent connections (e.g., via shared liquidity pools or repeated trading partners) could reveal more sophisticated manipulation strategies. Incorporating such intermediary structures into temporal graph modeling presents a promising direction for future research.

1) Data Preprocessing: The preprocessing of ERC-20 token transfer data involves several key steps to ensure data quality. First, duplicate records and entries with incomplete data are removed. Next, a logarithmic transformation is applied to the transaction value to mitigate the impact of extremely large transactions. Subsequently, transaction time is converted to Unix timestamp and shifted based on the time of the first transaction to enhance the consistency of temporal features. Finally, a transaction graph is constructed from the preprocessed data, where accounts are represented as nodes and transactions as edges, providing a solid foundation for precise analysis and PD detection.

2) Node Feature Extraction: Extracting relevant and efficient features from blockchain data is crucial for robust graph learning and the effective detection of PD schemes. We calculate the static and dynamic features for nodes and edges using NetworkX.[1] In detail, we extract the following features for each node:

- *Closeness centrality* measures the reciprocal of the average shortest path distance to all other nodes, indicating the node's accessibility within the network. High closeness centrality can signal central roles in transaction networks, often exploited in PD schemes.
- *Betweenness centrality* captures how frequently a node appears on the shortest paths between other nodes, reflecting its role as a critical intermediary. Nodes with high betweenness centrality may be orchestrating large-scale transactions typical of PD activities.
- *In-degree centrality* measures the number of incoming edges, showing the node's ability to attract transactions. A sudden increase in in-degree centrality could indicate coordinated buying during the pump phase.

[1]https://networkx.org/documentation/stable/reference/algorithms/centrality

- *Out-degree centrality* reflects the number of outgoing edges, representing the node's transaction initiation activity. High out-degree centrality might suggest a rapid sell-off during the dump phase.
- *Degree centrality* is the total number of connected edges, indicating a node's overall engagement in the network. Nodes with unusually high connectivity could be central to PD schemes.
- *Eigenvector centrality* assesses a node's influence based on the centrality of its neighbors, highlighting connections to highly influential nodes. High influence nodes may be key players in market manipulation.
- *Katz centrality* considers both direct and indirect connections, emphasizing the importance of short and long paths. This feature helps detect complex interaction patterns that could indicate coordinated market manipulation.
- *PageRank* measures a node's importance based on random walks through the network, adjusted for network structure. High pagerank could suggest a node's pivotal role in driving trading dynamics during a PD event.
- *Trading partner diversity* measures the variety of an account's trading partners, calculated as the ratio of unique partners to total transactions. Low diversity could signal a strategy to manipulate prices with a few accounts.

These features intuitively link to PD activities by highlighting nodes that demonstrate unusual influence, connectivity, and centrality, key indicators of the orchestrated behaviors typical in PD schemes in DeFi markets.

3) Edge Feature Extraction: For all transactions collected between two nodes, we extract the following key features to enhance graph learning:

- *Interaction count* measures the frequency of transactions between two nodes, highlighting regularity or sudden spikes that could indicate coordinated market manipulation.
- *Average transaction value* provides the mean transaction size between two nodes, helping to identify significant deviations that may signal abnormal trading behavior typical of PD schemes.
- *Transaction volatility* captures the variance in transaction values, reflecting the stability or fluctuation of trading amounts. High volatility can suggest erratic behavior, often seen during pump or dump phases.
- *Transaction magnitude* directly assesses the value of each transaction, crucial for detecting large, unusual transactions that are characteristic of PD activities.

These features are designed to capture key aspects of trading behavior that are often manipulated during PD schemes. Regular interactions might suggest an established trading relationship, while sudden spikes could indicate a coordinated effort to influence the market. Large deviations in average transaction value or high volatility may signal the inflow or outflow of capital in a manner consistent with PD strategies. The magnitude of individual transactions is critical for identifying outliers that could represent attempts to artificially inflate or deflate token prices. Collectively, these features enable robust detection of suspicious activities within DeFi markets, supporting the effectiveness of the PUMPWATCHER system in identifying and mitigating PD schemes.

Algorithm 1 Temporal Graph Sampling in PD-GNN

Input: the input graph g, list of numbers of neighbors to sample per edge type for each GNN layer $neig_{out}$, filter of the timestamp to sample ts, time of the transaction $time$, initial node set for the first layer neighbor sampling $neighbor$,
Output: list of sampled frontiers $frontiers$,
$frontiers \leftarrow [neighbor]$ // *Init frontiers list*
for $k \in neig_{out}$ **do**
  $g' \leftarrow \text{subgraph}(g, neighbor)$ // *Extract neighbor subgraph*
  $g'.\text{remove\_edges}(g[time] > ts)$ // *Time filter*
  $neighbor \leftarrow \text{select\_topk}(g', neighbor, g[time], k)$ // *Top-k sampling*
  $frontiers.\text{append}(neighbor)$ // *Store frontier*
return $frontiers$

C. PD-GNN

The objective is to enhance the detection of PD schemes in DeFi markets by learning transaction representations that incorporate temporal information. This approach models the dynamic nature of transactions, accounting for both their structural and temporal evolution. By integrating temporal features, the temporal PD-GNN can more effectively identify sudden and suspicious activities indicative of PD schemes. The process involves several key steps: temporal graph sampling, memory updating, time encoding, and node embedding.

1) Temporal Graph Sampling: The model begins by sampling subgraphs based on temporal information. Let $G_t = (V_t, E_t)$ represent the graph at a specific time t, where V_t is the set of nodes and E_t is the set of edges. Given a set of seed nodes S and a timestamp filter ts, the sampling process constructs a subgraph $G' = (V', E')$ by selecting nodes and edges before the timestamp filter ts from the most recent transaction involving any node in S. This approach ensures that the sampled subgraph accurately reflects the evolving nature of transactions over time.

Algorithm 1 details the temporal graph sampling procedure used in PD-GNN, which extracts subgraphs under temporal constraints to enable time-aware learning. The algorithm takes as input the graph g, a list $neig_{out}$ specifying the number of neighbors to sample at each GNN layer, a timestamp filter ts, the edge time attribute $time$, and an initial node set $neighbor$ (referred to as the seed nodes S above) for the first layer. For each value in $neig_{out}$ (corresponding to each GNN layer), the algorithm first extracts a subgraph g' containing the current set of neighbor nodes. It then removes from g' all edges whose $time$ attribute exceeds ts, ensuring temporal consistency. Next, for each node in the current neighbor set, the algorithm selects the top k neighbors based on the temporal information, where k is specified by the current value in $neig_{out}$. The newly sampled neighbors are stored as a frontier and appended to the frontiers list. This process is repeated for all GNN layers, and finally, the algorithm returns the list $frontiers$, which contains the

sampled neighbor sets for each layer, forming the basis for temporal subgraph construction in PD-GNN.

2) Memory Updating: The model maintains a memory vector $m_i(t)$ for each node i, updated with each transaction. For an interaction $e_{ij}(t)$ between nodes i and j at time t, the memory update function utilizes a Gated Recurrent Unit (GRU) as $m_i(t) = \text{GRU}(m_i(t^-), e_{ij}(t))$, where $m_i(t^-)$ represents the memory of node i before the interaction. This captures long-term dependencies and transaction patterns by storing historical behavior in memory vectors. For a new node, its memory is initialized to a zero vector and updated incrementally as new transactions occur.

3) Time Encoding: The model encodes the timestamp of each transaction using a finite Fourier series. The time encoding function $\Phi(t)$ is defined as:

$$\Phi(t) = [\cos(\omega_0 t + \psi_0), \cos(\omega_1 t + \psi_1), \ldots, \cos(\omega_n t + \psi_n)], \tag{1}$$

where ω and ψ are learnable parameters. This encoding captures periodic patterns and temporal dependencies in the transaction data. The use of a Fourier series allows the model to learn multi-scale temporal information, enabling the prediction and understanding of various temporal dynamics.

4) Message Passing and Aggregation: This step involves propagating information between nodes to update their representations. For an interaction event $e_{ij}(t)$ between a source node i and a target node j at time t, a message $msg(t)$ is generated as follows:

$$msg(t) = \text{concat}(m_i(t^-), m_j(t^-), e_{ij}(t), \Phi(t)). \tag{2}$$

This message updates the memory vectors of both the source and target nodes, capturing the directionality and context of the interaction.

The model also addresses the scenario where multiple events involve the same node within a batch by using an aggregation strategy. Given a set of messages $msg_i(t_1), \ldots, msg_i(t_b)$ for events involving node i occurring at times $t_1, \ldots, t_b \le t$, the aggregated message $\bar{msg}_i(t)$ is computed as:

$$\bar{msg}_i(t) = \max(msg_i(t_1), \ldots, msg_i(t_b)). \tag{3}$$

5) Node Embedding: The node embedding process generates time-aware embeddings by integrating temporal and structural features. For a node i at time t, the embedding $z_i(t)$ is computed by aggregating messages from its neighbors:

$$z_i(t) = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} \cdot \text{concat}\left(m_j(t), e_{ij}(t), \Phi\left(t - t_{ij}\right)\right)\right) \tag{4}$$

where σ denotes a non-linear activation function, $\mathcal{N}(i)$ is the set of neighbors of node i, and α_{ij} represents the attention weights, computed as:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}(i)} \exp(e_{ik})}. \tag{5}$$

This attention mechanism prioritizes more relevant transactions by assigning them higher importance in the embedding computation.

6) Edge Embedding: The edge embedding step updates the representations of edges using the newly computed node embeddings. For an edge $e_{ij}(t)$ between nodes i and j, where $z_i(t)$ and $z_j(t)$ denote the embeddings of nodes i and j at time t, the updated representation $h_{ij}(t)$ is computed as:

$$h_{ij}(t) = \text{concat}(z_i(t), z_j(t), e_{ij}(t)). \tag{6}$$

These enhanced edge representations, integrating temporal and structural information, improve the model's ability to detect PD schemes by capturing the evolving transaction dynamics and complex interactions within the DeFi market. The enhanced representations are then utilized in downstream tasks such as detecting PD activities.

D. CL-Based Representation Enhancer

The CL-based representation enhancer aims to refine node and edge representations using contrastive learning techniques. The core idea is to enhance these representations by maximizing the similarity between positive samples (transactions exhibiting normal behavior) and minimizing the similarity between negative samples (transactions associated with PD schemes). This contrastive learning approach effectively distinguishes between normal and anomalous activities, thereby improving the model's detection capabilities. The module integrates contrastive learning into the GNN framework, allowing the model to capture subtle differences in transaction patterns that may indicate fraudulent behavior. This process involves three main steps: sample creation, contrastive loss computation, and representation optimization.

1) Sample Creation: Positive and negative pairs are generated from the transaction graph $G = (V, E)$, where V and E represent the sets of nodes and edges, respectively. For each node i, positive pairs (i, j^+) are created by selecting neighboring nodes j^+ known or suspected to be involved in PD schemes. Negative pairs (i, j^-) are formed by selecting nodes j^- that demonstrate normal transactional behavior. This process ensures that the model learns to effectively differentiate between normal and anomalous behaviors.

2) Contrastive Loss Computation: The contrastive loss $\mathcal{L}$ for each pair is computed as follows. Given the embeddings z_i, z_{j^+}, and z_{j^-} for nodes i, j^+, and j^-, respectively, the loss function $\mathcal{L}$ is defined by:

$$\begin{aligned}\mathcal{L} = {} & W_P \cdot \left(\sum_{i \in S_P} \frac{1}{|S_P|^2} \sum_{j^+ \in S_P} \|z_i - z_{j^+}\|_2^2\right) \\ & + W_N \cdot \left(\sum_{i \in S_N} \frac{1}{|S_N|^2} \sum_{j^- \in S_N} \|z_i - z_{j^-}\|_2^2\right) \\ & + W_D \cdot \left(\sum_{j^+ \in S_P} \sum_{j^- \in S_N} \max\left(0, \text{margin} - \|z_{j^+} - z_{j^-}\|_2^2\right)\right),\end{aligned} \tag{7}$$

where S_P and S_N are the sets of positive and negative samples. W_P, W_N, and W_D are the weights assigned to the positive, negative, and differentiating loss components, respectively. Margin is a predefined threshold. $\|\cdot\|_2^2$ measures the Euclidean

distance between two vectors by computing the square root of the sum of the squared differences between their components. The first two terms of the contrastive loss minimize intra-class distances for positive and negative pairs, while the third enforces a margin between them, shaping the embedding space to clearly separate normal and PD-related behaviors. Gradients of the loss are computed via backpropagation and flow through the temporal PD-GNN and contrastive encoder, enabling updates to GNN weights, attention mechanisms, and memory functions. This end-to-end optimization, performed using adaptive moment estimation (Adam) or stochastic gradient descent (SGD), allows the model to learn discriminative temporal and structural features that generalize across diverse trading patterns.

3) Representation Optimization: The node and edge representations are updated by minimizing the contrastive loss. Embeddings z_i are optimized using SGD or other appropriate optimization algorithms. The update rule is given by:

$$z_i \leftarrow z_i - \eta \nabla_{z_i} \mathcal{L}, \tag{8}$$

where η is the learning rate, and $\nabla_{z_i} \mathcal{L}$ is the gradient of the contrastive loss with respect to the embedding z_i. This iterative process refines the embeddings, making them more discriminative for PD detection.

E. PD Detector

This module uses LightGBM, a gradient boosting framework developed by Microsoft, as the downstream classifier for detecting PD schemes. LightGBM is optimized for efficiency and accuracy, particularly in handling large-scale, high-dimensional datasets. It employs a histogram-based decision tree algorithm for faster training and low memory consumption, making it ideal for processing large datasets in memory-constrained environments. Additionally, LightGBM's optimized split algorithm and histogram approach enhance accuracy by reducing loss function values. The framework also supports parallel training, effectively utilizing multi-core CPUs, which is crucial for large-scale datasets. These attributes make LightGBM well-suited for the PD Detector module, ensuring efficient and accurate classification of transactions as either normal or PD-related.

IV. Performance Evaluation

In this section, we report performance of PumpWatcher.

A. Experimental Setup

1) Dataset: Our dataset, developed in collaboration with Uniswap [30] and MetaTrust [31], spans the one-month period from December 1 to 31, 2022. All ERC-20 token transactions were obtained from public sources using the Etherscan API, covering the full Ethereum ledger regardless of whether transactions directly involved Uniswap. After initial extraction, we performed rigorous data cleaning: we removed transactions containing null fields, special symbols (e.g., @, %, #) in token names that are often associated with fraudulent characteristics such as "high returns", or severe data-type mismatches (e.g., non-numeric values in amount fields). We then excluded all transactions involving scam tokens previously flagged by platforms such as Etherscan. From the remaining tokens, we selected those with a monthly transaction count between 300 and 3,000 to balance coverage and noise, resulting in 924,508 valid transactions across 858 tokens. Ground truth labels were generated using a heuristic-based labeling engine and validated with manual review by MetaTrust. The resulting graphs comprise 338,744 nodes and reflect both common and suspicious trading behaviors within the DeFi ecosystem.

TABLE I
Statistics of the Dataset

Dataset Split	Txs	Timeframe	Accounts	Tokens
Training	647,155 (70%)	21.23 days	18,539	682
Validation	138,676 (15%)	5.43 days	7,328	247
Test	138,677 (15%)	4.34 days	6,304	229
Total	924,508 (100%)	1 month	26,549	858

As shown in Table I, the dataset exhibits several key characteristics: (i) since each transaction involves two participating accounts, each account engages in approximately 69.65 ($924,508 \times 2 \div 26,549 \approx 69.65$) transactions across all tokens over the month, averaging two transactions per day; (ii) accounts interact with an average of 12.76 different tokens, reflecting a broad range of trading interests; (iii) despite an average of 5.46 transactions per token per account, our model is designed to detect anomalous behavior even with this relatively sparse data. For model training and evaluation, the dataset is temporally split, with 70% used for training, 15% for validation, and 15% for testing. This approach ensures the model is evaluated on future, unseen data, closely aligning with real-world application scenarios.

2) Metrics: We employ the following metrics to assess the performance of PumpWatcher in detecting DeFi PD schemes. False positive rate (FPR) measures the proportion of normal transactions incorrectly classified as PD-related. False negative rate (FNR) quantifies the proportion of PD-related transactions that are missed by the detector. F1 score provides a harmonic mean of precision and recall, reflecting the balance between detecting true positives and avoiding false positives. Additionally, we utilize the balanced accuracy (BAC) metric, which averages the sensitivity and specificity to account for class imbalance. To visually assess the performance, we also plot Receiver Operating Characteristic (ROC) curves, which illustrate the trade-off between the true positive rate and false positive rate at various threshold settings, providing a comprehensive view of the classifier's discriminative ability.

3) Implementation: Our model is trained over 30 epochs with a batch size of 512, using the Adam optimizer with a learning rate of 0.0001. We experimented with node embedding dimensions, memory dimensions, and temporal dimensions ranging from 16 to 256, and set the edge embedding dimension at 256 with an advanced edge embedding dimension of 275. We explored GRU and RNN for memory updating, utilized the latest message in sequence aggregation, and varied the number of neighbors (5, 10, 15, 20), sample

TABLE II
FPR, FNR, F1, AND BAC UNDER DIFFERENT EMBEDDING DIMENSIONS

Dim	FPR	FNR	F1	BAC
16	0.081	0.159	0.712	0.880
32	0.030	0.157	**0.828**	0.906
64	0.050	0.117	0.801	0.916
128	0.077	**0.077**	0.763	**0.923**
256	**0.009**	0.324	0.779	0.833

TABLE III
FPR, FNR, F1, AND BAC UNDER DIFFERENT NUMBERS OF NEIGHBORS

Neighbors	FPR	FNR	F1	BAC
5	0.028	0.253	0.775	0.859
10	0.077	**0.077**	0.763	**0.923**
15	**0.017**	0.224	**0.823**	0.879
20	0.019	0.277	0.783	0.852

methods (Top-k, Random), attention heads (4, 6, 8), and k-hop (1, 2, 3) to optimize temporal graph learning, with a feature and attention dropout of 0.6. In the Contrastive Learner, we adjusted the margin parameter, testing values from 0.5 to 100, to enhance the separation of positive and negative samples. The final classification is performed by a LightGBM-based Transaction Classifier, where we experimented with the number of leaves (63, 127, 255), learning rates (0.1, 0.2), feature fraction (0.9), bagging fraction (0.9), and bagging frequency (5) to ensure robust and efficient detection of PD schemes in DeFi markets.

B. Overall Performance

1) Impact Of Different Embedding Dimensions: For the evaluation of the temporal PD-GNN module, we fixed the number of nearest neighbors at 10, the number of attention heads at 4, the number of hops at 2, and used concatenation for node embedding generation, with top-k sampling for GNN and GRU for memory updating. We then varied the embedding dimension between 16, 32, 64, 128, and 256. Table II presents the results under these configurations. With an embedding dimension of 16, the model achieved an FPR of 0.081, an FNR of 0.159, an F1 score of 0.712, and a BAC of 0.880. Increasing the embedding dimension to 32 resulted in an FPR of 0.030, an FNR of 0.157, an F1 score of 0.828, and a BAC of 0.906. With an embedding dimension of 64, the FPR was 0.050, the FNR was 0.117, the F1 score was 0.801, and the BAC was 0.916. An embedding dimension of 128 resulted in an FPR of 0.077, an FNR of 0.077, an F1 score of 0.763, and a BAC of 0.923. Finally, with an embedding dimension of 256, the FPR decreased to 0.009, but the FNR increased to 0.324, resulting in an F1 score of 0.779 and a BAC of 0.833. These results indicate that while increasing the embedding dimension can enhance performance, excessively large dimensions may lead to overfitting and decreased generalization.

2) Impact of Different Number of Neighbors: For the evaluation of the temporal PD-GNN module with a fixed embedding dimension of 128, we varied the number of neighbors between 5, 10, 15, and 20. Table III presents the results under these configurations. With 5 neighbors, the model achieved an FPR of 0.028, an FNR of 0.253, an F1 score of 0.775, and a BAC of 0.859. Increasing to 10 neighbors improved generalization, reducing both FPR and FNR, and yielded the highest BAC of 0.923. Although further increasing to 15 and 20 neighbors reduced training error, we observed that validation loss began to increase, signaling overfitting to the training set. These results highlight the importance of balancing neighborhood size to maximize model expressiveness while preserving generalization.

TABLE IV
FPR, FNR, F1, AND BAC UNDER DIFFERENT NUMBERS OF ATTENTION HEADS

Heads	FPR	FNR	F1	BAC
4	**0.077**	**0.077**	**0.763**	**0.923**
6	0.128	0.183	0.618	0.844
8	0.125	0.199	0.615	0.838

TABLE V
FPR, FNR, F1, AND BAC UNDER DIFFERENT K-HOP GNNS

K-hop	FPR	FNR	F1	BAC
1	0.140	0.078	0.653	0.891
2	0.077	**0.077**	**0.763**	**0.923**
3	**0.073**	0.225	0.690	0.851

3) Impact of Different Number of Attention Heads: For the evaluation of the temporal PD-GNN module, we fixed the dimensionality at 128 and the number of neighbors at 10. We then varied the number of attention heads between 4, 6, and 8. Table IV presents the results under these configurations. With 4 attention heads, the model achieved an FPR of 0.077, an FNR of 0.077, an F1 score of 0.763, and a BAC of 0.923. Increasing the number of heads to 6 resulted in an FPR of 0.128, an FNR of 0.183, an F1 score of 0.618, and a BAC of 0.844. Using 8 heads, the FPR decreased to 0.125, the FNR was 0.199, the F1 score was 0.615, and the BAC was 0.838. These results suggest that 4 heads provide a better performance.

4) Impact of Different Number of GNN Layers: For the final evaluation of the temporal PD-GNN module, we fixed the dimensionality at 128, the number of neighbors at 10, and the number of attention heads at 4. We then varied the number of hops (k-hop) between 1, 2, and 3. Table V presents the results under these configurations. Using 1-hop, the model achieved an FPR of 0.140, an FNR of 0.078, an F1 score of 0.653, and a BAC of 0.891. Using 2-hop, the FPR was 0.077, the FNR was 0.077, the F1 score was 0.763, and the BAC was 0.923. With 3-hop, the FPR decreased to 0.073, the FNR increased to 0.225, the F1 score decreased to 0.690, and the BAC dropped to 0.851. These results indicate that the 2-hop configuration achieves the best balance between false positive rate and false negative rate.

5) Impact of Node Embedding Methods for Edge Construction: For the final evaluation of the temporal PD-GNN module, we fixed the dimensionality at 128, the number of neighbors at 10, the number of attention heads at 4, and the number of hops at 2. We then varied the method of

TABLE VI
PERFORMANCE UNDER DIFFERENT NODE EMBEDDING FOR EDGE EMBEDDING

Node Embedding Method	FPR	FNR	F1	BAC
Element-wise Addition	0.085	0.120	0.725	0.897
Concatenation	**0.077**	**0.077**	**0.763**	**0.923**
Element-wise Multiplication	0.148	0.093	0.634	0.879
Element-wise Absolute Difference	0.143	0.193	0.592	0.832

TABLE VII
PERFORMANCE UNDER DIFFERENT GNN SAMPLING

GNN Sampling	FPR	FNR	F1	BAC
Top-k	**0.077**	**0.077**	**0.763**	**0.923**
Random Sampling	0.184	0.255	0.508	0.780

TABLE VIII
PERFORMANCE UNDER DIFFERENT MEMORY UPDATE METHODS

Memory Update Method	FPR	FNR	F1	BAC
GRU	**0.077**	0.077	**0.763**	**0.923**
RNN	0.112	**0.076**	0.698	0.906

TABLE IX
PERFORMANCE UNDER DIFFERENT NUMBER OF LEAVES

Leaves	FPR	FNR	F1	BAC
63	0.076	0.175	0.713	0.874
127	0.077	**0.077**	**0.763**	**0.923**
255	**0.069**	0.175	0.727	0.878

TABLE X
IMPACT OF DIFFERENT LEARNING RATES IN LIGHTGBM

Learning Rate	FPR	FNR	F1	BAC
0.1	0.069	**0.175**	**0.727**	**0.878**
0.2	**0.066**	0.203	0.717	0.865

generating node embeddings for edge embedding between element-wise addition, concatenation, element-wise multiplication, and element-wise absolute difference. Table VI presents the results under these configurations. Using element-wise addition, the model achieved an FPR of 0.085, an FNR of 0.120, an F1 score of 0.725, and a BAC of 0.897. Using concatenation, the FPR was 0.077, the FNR was 0.077, the F1 score was 0.763, and the BAC was 0.923. With element-wise multiplication, the FPR increased to 0.148, the FNR rose to 0.093, the F1 score decreased to 0.634, and the BAC dropped to 0.879. Using element-wise absolute difference, the FPR was 0.143, the FNR was 0.193, the F1 score was 0.592, and the BAC was 0.832. These results indicate that the concatenation method is the most effective, achieving the lowest false positive rate and highest balanced accuracy.

6) Impact of Neighbour Sampling in PD-GNN: For the final evaluation, we fixed the dimensionality at 128, the number of neighbors at 10, the number of attention heads at 4, the number of hops at 2, and the method of generating node embeddings as concatenation. We then varied the GNN sampling method between top-k sampling and random sampling. Table VII presents the results under these configurations. With top-k sampling, the model achieved an FPR of 0.077, an FNR of 0.077, an F1 score of 0.763, and a BAC of 0.923. When using random sampling, the FPR increased to 0.184, the FNR rose to 0.255, the F1 score decreased to 0.508, and the BAC dropped to 0.780. These results indicate that top-k sampling is significantly more effective in reducing false positives and achieving a balanced accuracy, highlighting its superiority in this context.

7) Impact of Different Memory Updating Methods: For the final evaluation of the temporal PD-GNN module, we fixed the dimensionality at 128, the number of neighbors at 10, the number of attention heads at 4, the number of hops at 2, the method of generating node embeddings as concatenation, and the GNN sampling method as top-k. We then varied the memory update method between GRU and RNN. Table VIII presents the results under these configurations. Using the GRU method, the model achieved an FPR of 0.077, an FNR of 0.077, an F1 score of 0.763, and a BAC of 0.923. When using the RNN method, the FPR increased to 0.112, the FNR decreased slightly to 0.076, the F1 dropped to 0.698, and the BAC decreased to 0.906. These results indicate that while the RNN method slightly improves the FNR, the GRU method is more effective overall in maintaining a lower false positive rate and achieving a balanced accuracy.

8) Impact of Number of Leaves: For the evaluation of the PD Detector module, we tuned the LightGBM classifier by varying the number of leaves to 63, 127, and 255, respectively. Initially, the settings included a learning rate (lr) of 0.1 and Synthetic Minority Over-sampling Technique (SMOTE) sampling. Table IX presents the results under different numbers of leaves. With 63 leaves, the model achieved an FPR of 0.076, an FNR of 0.175, an F1 score of 0.713, and a BAC of 0.874. When increasing the number of leaves to 127, the FPR rose to 0.077, the FNR decreased to 0.077, the F1 score improved to 0.763, and the BAC improved to 0.923. At 255 leaves, the FPR was 0.069, the FNR increased to 0.175, the F1 score was 0.727, and the BAC was 0.878. These results indicate that while increasing the number of leaves can improve balanced accuracy, it may also lead to higher false positive rates and lower F1 scores, suggesting a trade-off between different performance metrics.

9) Impact of Learning Rate: For the next evaluation step, we fixed the number of leaves at 127 and varied the learning rate (lr) to 0.1 and 0.2. Table X presents the results under these configurations. With a learning rate of 0.1, the model achieved an FPR of 0.069, an FNR of 0.175, an F1 score of 0.727, and a BAC of 0.878. When the learning rate was increased to 0.2, the FPR slightly decreased to 0.066, but the FNR increased to 0.203, the F1 score dropped to 0.717, and the BAC decreased to 0.865. These results suggest that a lower learning rate results in a better balance of performance metrics, while a higher learning rate may increase the risk of misclassifications. Figure 4 presents the loss under different training epochs with lr as 0.1 for training LightGBM and 0.0001 for training PD-

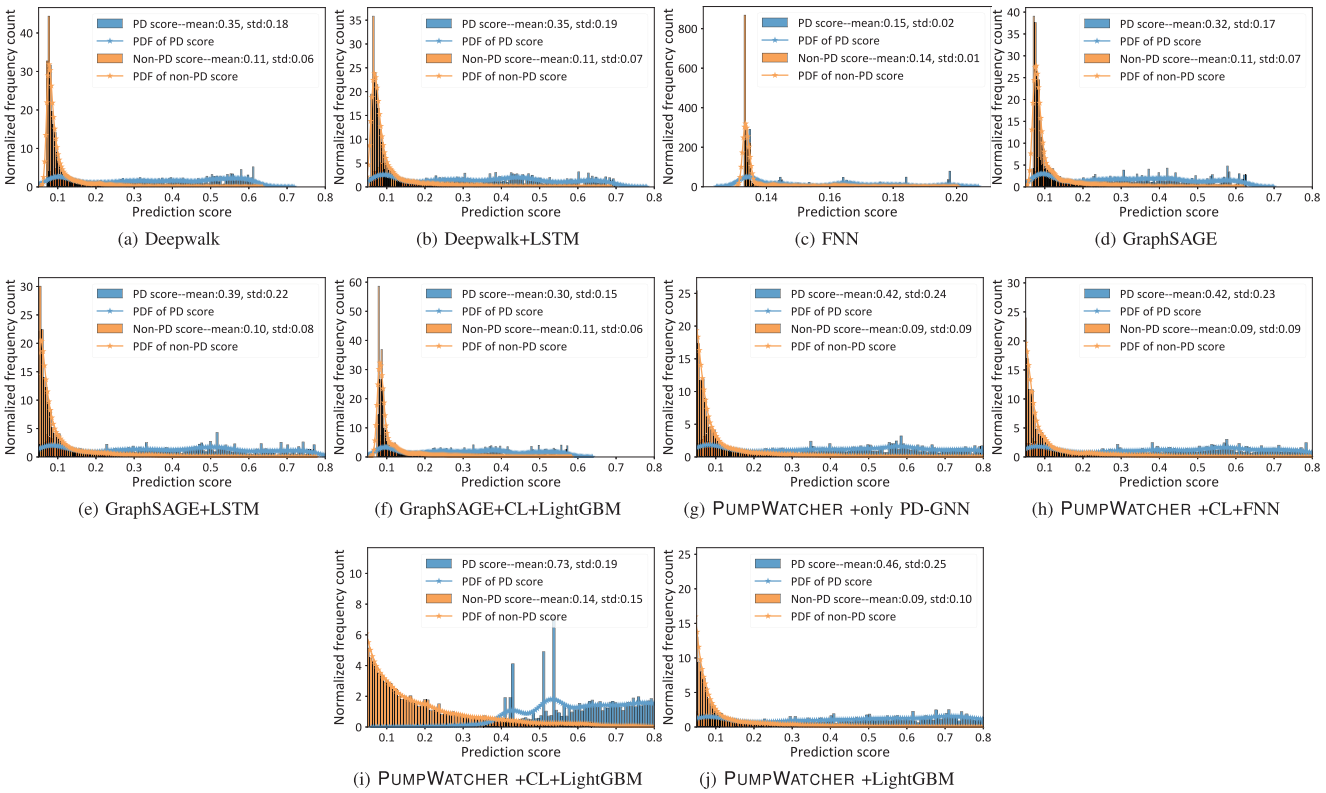


Fig. 3. Normalized frequency count and PDF of prediction score under different methods.

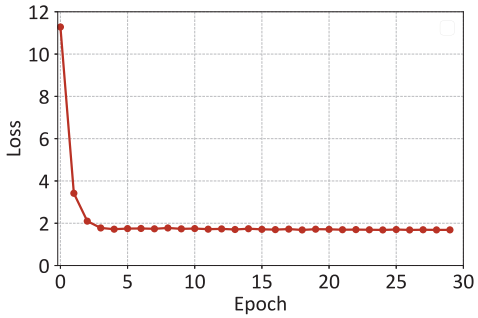


Fig. 4. Loss under different training epochs.

TABLE XI
PERFORMANCE UNDER DIFFERENT OVERSAMPLING METHODS

Oversampling Method	FPR	FNR	F1	BAC
SMOTE	**0.069**	**0.175**	**0.727**	**0.878**
Random Sampling	0.102	0.205	0.648	0.846

TABLE XII
PERFORMANCE UNDER DIFFERENT METHODS

Model	FPR	FNR	F1	BAC
Linear Regression	**0.006**	0.973	0.050	0.510
LSTM	0.322	0.527	0.267	0.575
GraphSage	0.998	**0.005**	0.236	0.498
GraphSage+LSTM	0.319	0.128	0.444	0.776
Our method	0.077	0.077	**0.763**	**0.923**

GNN. It exhibits that it achieves stable performance at the 5_{th} epoch.

10) Impact of Sampling Methods in Training PD Detector: For the final evaluation of the PD detector module, we fixed the number of leaves at 127 and the learning rate at 0.1, and then varied the oversampling method between SMOTE and random sampling. Table XI presents the results under these configurations. Using SMOTE sampling, the model achieved an FPR of 0.069, an FNR of 0.175, an F1 score of 0.727, and a BAC of 0.878. With random sampling, the FPR increased to 0.102, the FNR rose to 0.205, the F1 score decreased to 0.648, and the BAC decreased to 0.846. These results indicate that while random sampling slightly improves the F1 score, SMOTE sampling is more effective overall in reducing false positives and achieving a balanced accuracy.

C. Comparing With State-of-the-Art (SotA) Methods

For the comparative evaluation, we assessed the performance of our proposed method against several existing methods, including Linear Regression, LSTM, GraphSage, and GraphSage combined with LSTM. Table XII presents the results of this comparison. Figure 3 presents detailed normalized frequency count and probability density function (PDF) of prediction score under different methods. Linear Regression achieved an FPR of 0.006, an FNR of 0.973, an F1 score of 0.050, and a BAC of 0.510, indicating its limited ability to detect PD events accurately. The LSTM model showed an FPR of 0.322, an FNR of 0.527, an F1 score of 0.267, and a BAC of 0.575, reflecting moderate performance. GraphSage alone had an FPR of 0.998, an FNR of 0.005,

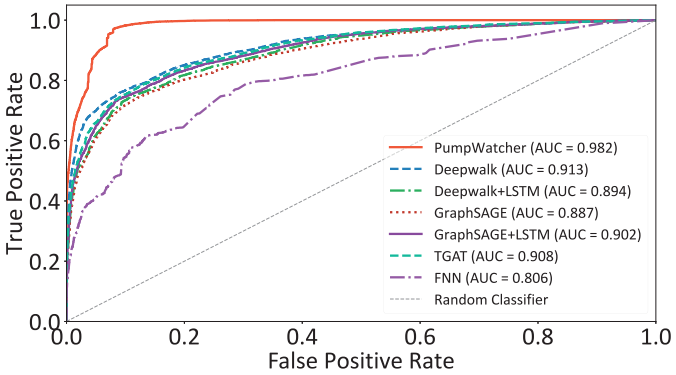


Fig. 5. ROC curves under different methods.

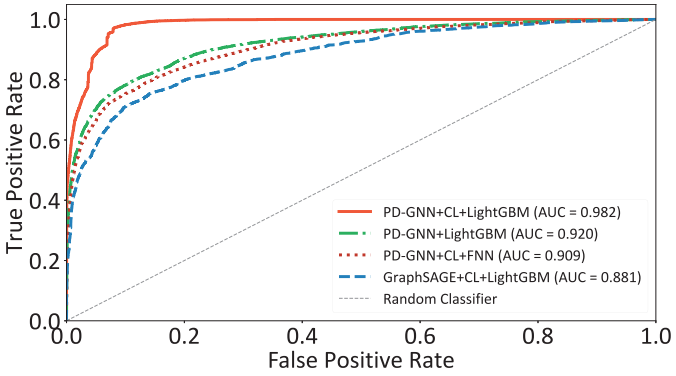


Fig. 6. ROC curves under different modules.

an F1 score of 0.236, and a BAC of 0.498, which indicates poor overall performance. The combination of GraphSage and LSTM improved the results, with an FPR of 0.319, an FNR of 0.128, an F1 score of 0.444, and a BAC of 0.776. Our method, however, outperformed all these models with an FPR of 0.077, an FNR of 0.077, an F1 score of 0.763, and a BAC of 0.923. These results demonstrate the superior effectiveness and robustness of our method in detecting PD schemes in DeFi markets. Figures 5 and 6 present ROC curves under different methods. PUMPWATCHER achieved the highest AUC of 0.982.

D. Real-World Performance

We also implemented PUMPWATCHER as a prototype, and deployed it in MetaTrust to detect PD schemes on Ethereum from August 1 to 20, 2024. During this period, it successfully identified 7,562 PD transactions involving 181 ERC-20 tokens. Of these, 2,198 PD schemes were detected proactively before any external reports, with subsequent confirmations through transaction analysis and market activity. Additionally, it flagged 9,442 transactions as suspicious, pending further investigation, with initial validations by blockchain auditors indicating a high probability of PD activity. Notably, PUMPWATCHER contributed to the prevention of potential investor losses estimated at $ 75,200. All detected PD schemes were responsibly disclosed to Web3 security communities, such as MetaTrust and ChainUp, to enhance market transparency and protection.

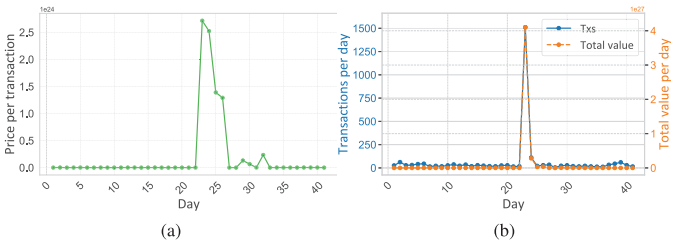


Fig. 7. GAMMA token: average price per transaction (a), average number of transaction and total value per day (b), during the period from November 21 to December 31, 2022.

E. Overhead Estimation

To evaluate deployment feasibility, we analyzed the computational costs of both training and inference phases. During training, the temporal GNN with contrastive learning required 12,822.61 seconds (3h 33min) and consumed 1,534.05 MB of GPU memory, while the downstream LightGBM classifier trained in just 3.19 seconds with 85.90 MB of CPU memory. In the inference phase, PUMPWATCHER predicted transaction labels in 55.72 seconds using 11.57 MB of GPU memory, and classification completed in 0.01 seconds with only 0.20 MB of CPU memory. These results demonstrate that PUMPWATCHER is computationally efficient and well-suited for real-world deployment.

F. Case Study

1) Case 1-GAMMA: In the ERC-20 market, the existence of tokens with similar symbols, such as GAMMA, GAMMA1, GAMMA2 raises concerns about potential market manipulation, where GAMMA could be used to disrupt the trading of GAMMA1 and GAMMA2 or intentionally create confusion. As illustrated in Figure 7, during the period from November 21 to December 31, 2022, GAMMA exhibited abnormal trading behavior, particularly on December 13, 2022, when both the number of transactions and the total trading volume spiked significantly compared to previous days. On December 14, 2022, the average transaction value remained unusually high before returning to normal levels on December 15, 2022. This pattern aligns with the typical characteristics of a PD scheme, where a token experiences a short-term surge in trading activity followed by a rapid return to pre-PD levels. Further analysis of the transactions on December 13, 2022, revealed an exceptionally anomalous address 0xb5...72, which is a contract address that engaged in a large number of transactions exchanging Wrapped Ether for GAMMA. Out of 1,441 total transactions for GAMMA that day, 1,194 (82.86%) involved this contract, suggesting a coordinated effort to convert the pumped GAMMA into the more stable Wrapped Ether, likely to profit from a PD event.

2) Case 2-Gifto: In the ERC-20 market, the token Gifto, shares the same symbol as another token, Gifto1. This similarity may have caused confusion or disruption in the market. As illustrated in Figure 8, Gifto exhibited a significant spike in the number of transactions, total trading volume, and average transaction value—patterns commonly associated with PD events.

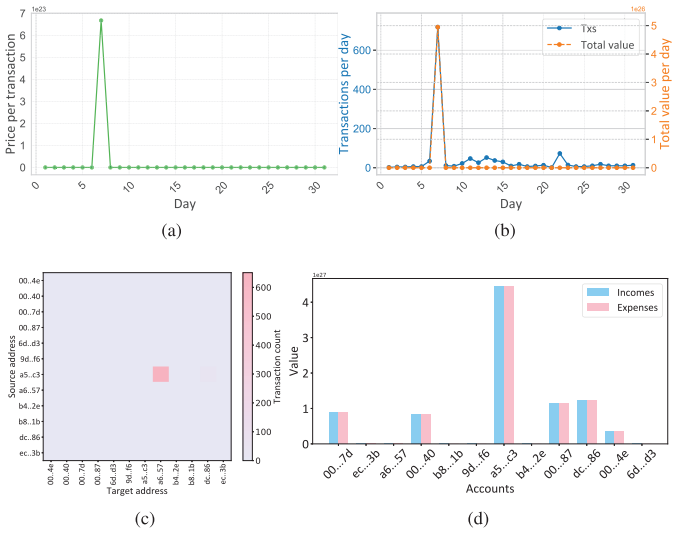


Fig. 8. `Gifto` token: average price per transaction (a), average number of transaction and total value per day (b), transaction matrix (c), incoming and outgoing fund flow (d), during the period from December 1 to December 31, 2022.

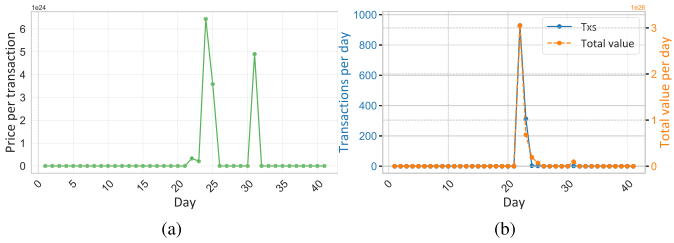


Fig. 9. `Sure` token: average price per transaction (a), average number of transaction and total value per day (b), during the period from November 21 to December 31, 2022.

A closer examination of the transactions on that day revealed unusual high-frequency trading between specific addresses, particularly between the contract address `0xa5...c3` and the account addresses `0xa6..57` and `0xdc..86`. Out of 717 transactions, 651 (90.79%) occurred between the first pair, and 43 (6.56%) between the second. Moreover, these addresses showed nearly balanced inflow and outflow, suggesting an orchestrated attempt at manipulating the market. Given the transaction patterns and the number of involved addresses, it is plausible that a PD attempt was made on December 7, 2022, though it may have been unsuccessful.

3) Case 3-`Sure`: On December 12, 2022, the ERC-20 token `Sure` exhibited a significant deviation from its usual trading behavior, with a sharp increase in both the number of transactions and the total trading volume. As illustrated in Figure 9, this spike aligns with typical PD patterns, where tokens experience a sudden surge in activity, followed by a rapid return to normal levels. Over the next few days, from December 13 to 15, 2022, the average transaction value remained abnormally high, before normalizing on December 16, 2022. After that, only a few transactions occurred for this token. In addition, we observed a large number of transactions with a value of zero after between addresses and the token contract December 12, 2022. This pattern is consistent with known PD events, where abnormal trading behavior temporarily persists before stabilizing. A detailed analysis of the transactions on December 12 revealed an address, `0x03..2e`, exhibiting highly irregular trading activity. Upon investigation via Etherscan, this address was identified as a contract facilitating exchanges between the `Sure` token and Wrapped Ether. Of the 1,587 total transactions associated with this contract, 1,574 occurred between December 12 and 13, 2022, with only 13 transactions happening thereafter, strongly indicating involvement in a PD scheme. This suggests that multiple accounts may have exploited the contract to convert the inflated `Sure` tokens into the more stable Wrapped Ether, profiting from the PD event.

V. Related Work

PD schemes in cryptocurrency markets have garnered significant attention due to their impact on market integrity and investor trust. Morgia et al. [9] conducted a comprehensive analysis of PD manipulations, focusing on the identification and detection of such schemes within cryptocurrency markets. Their research employs a combination of machine learning techniques and statistical methods to analyze trading patterns and social media signals, thereby providing a robust framework for real-time detection of these manipulative activities. Similarly, Victor et al. [14] explored the quantification and detection of PD schemes using market and social signals, emphasizing the importance of integrating various data sources to improve detection accuracy. Rajaei et al. [32] provided a comprehensive survey on PD detection using machine learning techniques, categorizing existing approaches based on the type of data utilized, such as market data and social media data, and discussing the strengths and weaknesses of each method.

Further advancements in PD detection have been made by researchers utilizing deep learning approaches. Chadalapaka et al. [16] proposed a novel deep learning-based framework for detecting cryptocurrency PD schemes. Their model leverages convolutional and recurrent neural networks to capture intricate patterns in trading data, demonstrating significant improvements in detection performance over traditional methods. In another study, Chen et al. [15] introduced an improved apriori algorithm to detect PD activities, highlighting the effectiveness of association rule mining in uncovering fraudulent trading behaviors. Xu et al. [11] presented an in-depth analysis of PD activities organized in Telegram channels, proposing several machine learning models, including random forest and generalized linear models, to predict the likelihood of a cryptocurrency to be pumped. Their study emphasized the importance of feature selection and data balancing to improve model performance.

Additionally, hybrid detection models have shown promise in enhancing the robustness and reliability of PD detection systems. Mansourifar et al. [33] developed a hybrid detection approach that combines supervised and unsupervised learning techniques to identify PD schemes. This hybrid model incorporates both historical trading data and real-time social media analysis, offering a comprehensive solution to detect

TABLE XIII

HIGH-LEVEL COMPARISON WITH EXISTING WORKS. TB: TEMPORAL-BASED. TG: TRANSACTION GRANULAR INDICATES IF IT ONLY CAN DETECT THE IN. MT: MULTIPLE TOKENS. ED: EARLY-DETECTING. PDA: PUBLIC DATASET

Method	Model	TB	TG	MT	ED	PDA
Kamps et al. [35]	Anomaly detection	○	○	●	○	●
Chen et al. [15]	Apriori	◐	○	○	○	○
Chadalapaka et al. [16]	Anomaly Transformer	◐	○	●	○	●
Xu et al. [11]	RF+GLM	◐	○	○	○	●
Bello et al. [34]	RF+AdaBoost	◐	○	●	○	○
Mirtaheri et al. [36]	SVM	○	○	●	○	●
Morgia et al. [9]	RF+LR	○	○	●	○	○
PUMPWATCHER (Ours)	PD-GNN + contrastive learning	●	●	●	●	●

and mitigate the impact of PD activities. Bello et al. [34] proposed a low-latency detection solution that aims to thwart cryptocurrency PD schemes by employing a real-time monitoring system integrated with machine learning algorithms. Morgia et al. [9] suggested using rush orders and other market features for early detection of PD events, employing random forest and logistic regression models to enhance detection capabilities. Victor et al. [14] developed an XGBoost model to detect PD events.

PUMPWATCHER *vs. Previous Methods:* Table XIII compares PUMPWATCHER with several state-of-the-art PD detection methods across key factors: temporal-based detection (TB), transaction granularity (TG), multiple token support (MT), early detection (ED), and public datasets (PDA). While previous methods, such as those by Kamps et al. [35] and Chen et al. [15], focus on post-event analysis and lack transaction-level granularity, PUMPWATCHER leverages temporal GNNs to perform early detection and capture fine-grained transactional patterns. Chadalapaka et al. [16] integrate some temporal dynamics but fail to achieve the same transaction-level precision, while Xu et al. [11] and Bello et al. [34] focus on token support without incorporating real-time, granular detection. Other approaches, like those by Mirtaheri et al. [36] and Morgia et al. [9], rely on delayed price fluctuations, which hinder early detection. In contrast, PUMPWATCHER combines a PD-GNN with contrastive learning to provide a real-time, robust solution, addressing these challenges and excelling in both transaction granularity and temporal analysis.

VI. DISCUSSION

Beyond its application to DeFi, the design of PUMPWATCHER is applicable to a range of domains involving time-sensitive, graph-structured interactions. For example, in financial fraud detection or anti-money laundering, where transaction graphs evolve rapidly, the same temporal GNN and contrastive learning mechanisms can be employed to identify suspicious transfers. Similarly, in e-commerce platforms or communication networks, transaction-level embeddings that capture both structural roles and temporal shifts can be leveraged to detect anomalous behaviors. PUMPWATCHER serves as a generalizable framework for real-time behavioral anomaly detection, applicable across decentralized or complex digital ecosystems, providing a meaningful contribution to security in dynamic, high-stakes environments.

Despite the utility of existing heuristic-based approaches for detecting PD schemes in DeFi, these methods typically rely on predefined thresholds and event-level patterns, such as sudden volume surges or sharp price changes. While effective in identifying large-scale, coordinated manipulations, they often fall short in detecting nuanced, small-scale, or failed PD attempts, particularly those engineered to bypass common detection rules. Moreover, heuristic methods generally identify suspicious events first, then retroactively label associated transactions, making them less suited for real-time detection. In contrast, our graph-based method leverages temporal and structural transaction features at the individual account level, allowing PUMPWATCHER to detect both prominent and subtle PD behaviors without requiring an explicit event trigger. This transaction-level granularity is especially important in decentralized markets where actors can use multiple pseudo-anonymous accounts and adapt rapidly to evade rule-based systems.

Regarding the dataset scope, our one-month collection from December 2022 was selected to strike a balance between diversity and manageability. This period includes multiple confirmed PD events, verified by both Uniswap and MetaTrust, and captures a wide range of trading behaviors, from high-frequency bursts to stealthy, unsuccessful manipulation attempts. It also reflects realistic market volatility, including low-liquidity token launches that are particularly vulnerable to exploitation. While the current timeframe supports rigorous evaluation and benchmarking, we acknowledge the value of broader temporal coverage. As part of future work, we plan to expand our dataset to span several months or quarters to capture seasonal variations, long-term behavioral shifts, and more diverse manipulation strategies. This will further validate the robustness and adaptability of PUMPWATCHER in evolving DeFi environments.

Beyond detecting transaction volume anomalies, the temporal PD-GNN is designed to identify structural and behavioral irregularities that characterize pump-and-dump activities. These include sudden bursts of connectivity (e.g., a dormant node initiating multiple high-value transfers within a short window), emergence of densely connected subgraphs indicative of coordinated behavior, and abrupt shifts in transaction patterns among tightly coupled wallets. By maintaining temporal memory for each node and encoding event timestamps, the model captures these transient topological changes over time. This enables PUMPWATCHER to flag suspicious behavior based on evolving graph structures, rather than relying solely on transaction volume spikes or static thresholds.

While contrastive learning enhances the robustness of transaction representations, we acknowledge the potential risk of overfitting to recurring patterns in specific tokens [37]. To mitigate this, we partition graphs on a per-token basis and use temporal augmentations to enforce representation invariance across different market phases. However, we note that the generalization capability to completely unseen tokens with unique trading dynamics warrants further evaluation. In future work, we plan to perform cross-token and cross-platform testing to quantify how well the learned features transfer to novel DeFi contexts.

GNNs enable fine-grained modeling of transactional behavior by encoding each account's historical activity and its evolving relationships within the trading network [37]. This graph-based structure aligns naturally with DeFi ecosystems, where wallets and token transfers correspond to nodes and edges, allowing the model to capture direct and indirect interactions. Such representations are particularly effective in detecting pseudo-cyclic trading patterns, often used by malicious actors controlling multiple wallets. The temporal component of GNNs allows the model to reason over time-sensitive behaviors, supporting real-time detection without reliance on post-event signals. This results in more precise, early identification of PD activities, strengthening market integrity and user protection. Additionally, large language models (LLMs) hold promise in enhancing PD detection by contextualizing unstructured data sources such as social media and token descriptions [38], [39], [40], [41], [42]. Combining GNNs for behavioral modeling with LLMs for intent and sentiment analysis could offer a powerful multimodal approach to proactive manipulation detection.

VII. Conclusion

This paper presents PumpWatcher, a graph-based framework for detecting PD schemes in DeFi by modeling ERC-20 token transactions as temporal graphs. Leveraging temporal GNNs and contrastive learning, it captures both structural and temporal dependencies critical for identifying coordinated and evasive manipulation patterns. Extensive experiments demonstrate that it significantly outperforms state-of-the-art baselines in accuracy and robustness. Through adaptive graph construction, feature-enhanced node and edge encoding, and efficient downstream classification, it delivers reliable PD detection across volatile token markets.

Acknowledgment

Any opinions, findings, conclusions, or recommendations expressed in this article are those of the authors and do not reflect the views of the National Research Foundation Singapore or the Cyber Security Agency of Singapore.

References

[1] A. P. Kalapaaking, I. Khalil, X. Yi, K.-Y. Lam, G.-B. Huang, and N. Wang, "Auditable and verifiable federated learning based on blockchain-enabled decentralization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 1, pp. 102–115, Jan. 2025.

[2] H. Ma, S. Huang, J. Guo, K.-Y. Lam, and T. Yang, "Blockchain-based privacy-preserving federated learning for mobile crowdsourcing," *IEEE Internet Things J.*, vol. 11, no. 8, pp. 13884–13899, Apr. 2024.

[3] M. B. Mollah et al., "Blockchain for the Internet of Vehicles towards intelligent transportation systems: A survey," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4157–4185, Mar. 2021.

[4] J. Xu, Y. Feng, D. Perez, and B. Livshits, "Auto.Gov: Learning-based governance for decentralized finance (DeFi)," *IEEE Trans. Services Comput.*, vol. 18, no. 3, pp. 1278–1292, May 2025.

[5] J. I. Ibañez, C. N. Bayer, P. Tasca, and J. Xu, "Triple-entry accounting, blockchain, and next of kin: Towards a standardisation of ledger terminology," in *Digital Assets*, 2025, pp. 198–226.

[6] Y. Luo, Y. Feng, J. Xu, and P. Tasca, "Piercing the veil of TVL: DeFi reappraised," in *Proc. Int. Conf. Financial Cryptography Data Secur. (FC)*, 2025.

[7] M. Xie et al., "DeFort: Automatic detection and analysis of price manipulation attacks in DeFi applications," in *Proc. 33rd ACM SIGSOFT Int. Symp. Softw. Test. Anal.*, Sep. 2024, pp. 402–414.

[8] J. Xu, K. Paruch, S. Cousaert, and Y. Feng, "SoK: Decentralized exchanges (DEX) with automated market maker (AMM) protocols," *ACM Comput. Surveys*, vol. 55, no. 11, pp. 1–50, Nov. 2023.

[9] M. La Morgia, A. Mei, F. Sassi, and J. Stefa, "The doge of wall street: Analysis and detection of pump and dump cryptocurrency manipulations," *ACM Trans. Internet Technol.*, vol. 23, no. 1, pp. 1–28, Feb. 2023.

[10] H. Fu, Y. Feng, C. Wu, and J. Xu, "Perseus: Tracing the masterminds behind cryptocurrency pump-and-dump schemes," 2025, *arXiv:2503.01686*.

[11] J. Xu and B. Livshits, "The anatomy of a cryptocurrency pump-and-dump scheme," in *Proc. USENIX Secur. Symp.*, 2018, pp. 1609–1625.

[12] W. Chen, T. Zhang, Z. Chen, Z. Zheng, and Y. Lu, "Traveling the token world: A graph analysis of Ethereum ERC20 token ecosystem," in *Proc. Web Conf.*, Apr. 2020, pp. 1411–1421.

[13] W. Li, J. Bu, X. Li, and X. Chen, "Security analysis of DeFi: Vulnerabilities, attacks and advances," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Aug. 2022, pp. 488–493.

[14] F. Victor and T. Hagemann, "Cryptocurrency pump and dump schemes: Quantification and detection," in *Proc. Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2019, pp. 244–251.

[15] W. Chen, Y. Xu, Z. Zheng, Y. Zhou, J. E. Yang, and J. Bian, "Detecting 'Pump & dump Schemes' on cryptocurrency market using an improved apriori algorithm," in *Proc. IEEE Int. Conf. Service-Oriented Syst. Eng. (SOSE)*, Apr. 2019, pp. 293–298.

[16] V. Chadalapaka, K. Chang, G. Mahajan, and A. Vasil, "Crypto pump and dump detection via deep learning techniques," 2022, *arXiv:2205.04646*.

[17] A. Webb, "Decentralized finance (DeFi) and its implications on traditional network economics: A comparative study on market power, pricing dynamics, and user adoption," *Int. J. Cryptocurrency Res.*, vol. 4, no. 1, pp. 40–46, Jun. 2024.

[18] S. Siby, U. Iqbal, S. Englehardt, Z. Shafiq, and C. Troncoso, "WebGraph: Capturing advertising and tracking information flows for robust blocking," in *Proc. USENIX Secur. Symp.*, 2021, pp. 2875–2892.

[19] Z. Xu, P. Fang, C. Liu, X. Xiao, Y. Wen, and D. Meng, "DEPCOMM: Graph summarization on system audit logs for attack investigation," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2022, pp. 540–557.

[20] Z. Yang, W. Pei, M. Chen, and C. Yue, "WTAGRAPH: Web tracking and advertising detection using graph neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2022, pp. 1540–1557.

[21] W. H. Cruz, F. Dahi, Y. Feng, J. Xu, A. Malhotra, and P. Tasca, "Amm-based dex on the xrp ledger," in *Proc. Int. Conf. Blockchain Cryptocurrency (ICBC)*, 2025, pp. 1–10.

[22] R. Liang et al., "Ponziguard: Detecting Ponzi schemes on Ethereum with contract runtime behavior graph," in *Proc. IEEE/ACM Int. Conf. Softw. Eng. (ICSE)*, Oct. 2024, pp. 755–766.

[23] F. Liang, C. Qian, W. Yu, D. Griffith, and N. Golmie, "Survey of graph neural networks and applications," *Wireless Commun. Mobile Comput.*, vol. 2022, pp. 1–18, Jul. 2022.

[24] W. Shi and R. Rajkumar, "Point-GNN: Graph neural network for 3D object detection in a point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1708–1716.

[25] I. Kumar, Y. Hu, and Y. Zhang, "EFLEC: Efficient feature-leakage correction in GNN based recommendation systems," in *Proc. 45th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2022, pp. 1885–1889.

[26] Y. Zhang, S. Gao, J. Pei, and H. Huang, "Improving social network embedding via new second-order continuous graph neural networks," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2022, pp. 2515–2523.

[27] B.-Y. Jin, S.-C. Li, and J.-W. Huang, "GraphSAGE-based spammer detection using social attribute relationship," in *Proc. Int. Conf. Technol. Appl. Artif. Intell.*, 2024, pp. 300–313.

[28] M. Sun, J. Xing, H. Wang, B. Chen, and J. Zhou, "MoCL: Data-driven molecular fingerprint via knowledge-aware contrastive learning from molecular graph," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 3585–3594.

[29] S. Chen et al., "MD-GNN: A mechanism-data-driven graph neural network for molecular properties prediction and new material discovery," *J. Mol. Graph. Model.*, vol. 123, Sep. 2023, Art. no. 108506.

[30] *Uniswap*. Accessed: Jul. 30, 2025. [Online]. Available: https://uniswap.org/security

[31] *Metatrust*. Accessed: Jul. 30, 2025. [Online]. Available: https://metatrust.io

[32] M. J. Rajaei and Q. H. Mahmoud, "A survey on pump and dump detection in the cryptocurrency market using machine learning," *Future Internet*, vol. 15, no. 8, p. 267, Aug. 2023.

[33] H. Mansourifar, L. Chen, and W. Shi, "Hybrid cryptocurrency pump and dump detection," 2020, *arXiv:2003.06551*.

[34] A. S. Bello, J. Schneider, and R. Di Pietro, "LLD: A low latency detection solution to thwart cryptocurrency pump & dumps," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, May 2023, pp. 1–9.

[35] J. Kamps and B. Kleinberg, "To the moon: Defining and detecting cryptocurrency pump-and-dumps," *Crime Sci.*, vol. 7, no. 1, pp. 1–18, Dec. 2018.

[36] M. Mirtaheri, S. Abu-El-Haija, F. Morstatter, G. V. Steeg, and A. Galstyan, "Identifying and analyzing cryptocurrency manipulations in social media," *IEEE Trans. Computat. Social Syst.*, vol. 8, no. 3, pp. 607–617, Jun. 2021.

[37] C. Wu et al., "TokenScout: Early detection of Ethereum scam tokens via temporal graph learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Dec. 2024, pp. 956–970.

[38] W. Sun et al., "Source code summarization in the era of large language models," in *Proc. IEEE/ACM 47th Int. Conf. Softw. Eng. (ICSE)*, Apr. 2025, pp. 1–13.

[39] W. Sun, X. Wang, H. Wu, D. Duan, Z. Sun, and Z. Chen, "MAF: Method-anchored test fragmentation for test code plagiarism detection," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng., Softw. Eng. Educ. Training (ICSE-SEET)*, May 2019, pp. 110–120.

[40] G. Tao, W. Sun, T. Han, C. Fang, and X. Zhang, "RULER: Discriminative and iterative adversarial training for deep neural network fairness," in *Proc. 30th ACM Joint Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Nov. 2022, pp. 1173–1184.

[41] C. Fang et al., "Esale: Enhancing code-summary alignment learning for source code summarization," *IEEE Trans. Softw. Eng.*, vol. 50, no. 8, pp. 2077–2095, Aug. 2024.

[42] K. Wang et al., "A comprehensive survey in LLM(-Agent) full stack safety: Data, training and deployment," 2025, *arXiv:2504.15585*.