# SIEGE: Self-Supervised Incremental Deep Graph Learning for Ethereum Phishing Scam Detection

Shucheng Li
National Key Lab for Novel Software Technology, Nanjing University
Nanjing, Jiangsu, China
shuchengli@smail.nju.edu.cn

Runchuan Wang
National Key Lab for Novel Software Technology, Nanjing University
Nanjing, Jiangsu, China
njucs_san@smail.nju.edu.cn

Hao Wu
National Key Lab for Novel Software Technology, Nanjing University
Nanjing, Jiangsu, China
hao.wu@nju.edu.cn

Sheng Zhong
National Key Lab for Novel Software Technology, Nanjing University
Nanjing, Jiangsu, China
zhongsheng@nju.edu.cn

Fengyuan Xu*
National Key Lab for Novel Software Technology, Nanjing University
Nanjing, Jiangsu, China
fengyuan.xu@nju.edu.cn

## ABSTRACT

The phishing scams pose a serious threat to the ecosystem of Ethereum which is one of the largest blockchains in the world. Such a type of cyberattack recently has caused losses of millions of dollars. In this paper, we propose a **S**elf-supervised **I**ncr**E**mental deep **G**raph l**E**arning (**SIEGE**) model, for the phishing scam detection problem on Ethereum. To overcome the data scalability challenge, we propose splitting the original Ethereum transaction data and constructing transaction graphs for each split. Confronted with the minimal labeled data available, we resort to graph-based self-supervised learning. We design a spatial pretext task to learn high-quality node embeddings inside a single graph split, as well as an incremental learning paradigm and a temporal pretext task to facilitate information flow between different graph splits. To evaluate the effectiveness of SIEGE, we gather a real-world dataset consisting of six-month Ethereum transaction records. The results demonstrate that our model consistently outperforms baseline approaches in both transductive and inductive settings.

## CCS CONCEPTS

• **Applied computing** → *Digital cash*; • **Computing methodologies** → *Artificial intelligence.*

## KEYWORDS

phishing scam detection, graph neural network, self-supervised learning

**ACM Reference Format:**
Shucheng Li, Runchuan Wang, Hao Wu, Sheng Zhong, and Fengyuan Xu. 2023. SIEGE: Self-Supervised Incremental Deep Graph Learning for Ethereum Phishing Scam Detection. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23), October 29-November 3, 2023, Ottawa, ON, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3581783.3612461

*Corresponding author.


*MM '23, October 29-November 3, 2023, Ottawa, ON, Canada*

ACM ISBN 979-8-4007-0108-5/23/10…$15.00
https://doi.org/10.1145/3581783.3612461

## 1 INTRODUCTION

Ethereum [49], one of the most popular and scalable blockchains, has reached $193 billion market cap so far. It unleashes the full potential of smart contracts, which automatically manage and approve transactions without a centralized entity, and thus attracts a lot of attention and resources via its various decentralized finance (DeFi) applications atop smart contracts. Also, more and more people are starting to believe that an open, trust-free blockchain like Ethereum is better suited as a "back end" of web 3.0. And some problems caused by centralized data storage, such as data leakage, could be better solved with Ethereum.

However, the thriving of Ethereum also puts millions of its users at risk of various malicious attacks [20], such as the phishing scam [5, 8, 25, 47, 50, 51], Ponzi scheme [1, 9], and money laundering [48]. Among these attack-related crimes, phishing scam is the category involving the largest amount of money[1]. Compared to traditional phishing scams via phishing emails, web services, and malware, phishing scams in Ethereum are more difficult to detect. This is because victims typically make transfers directly to malicious wallet accounts on Ethereum. Therefore, existing security analysis of software or smart contracts may not be sufficient to detect phishing scams on Ethereum. We head to transactions for opportunities to address the phishing scams faced by Ethereum.

Naturally, Ethereum transactions can be formulated into a large-scale evolving graph. There are many successful cases showing that deep graph learning technology can be applied to non-euclidean data like graph [16, 17, 24, 35, 42, 45]. However, two challenges prevent us from leveraging deep graph learning on phishing scam detection on Ethereum, as follows,

- **Data scalability**. Currently, there have been more than 1.7 billion transactions on Ethereum, and new ones are being added consistently. Thus the Ethereum transaction graph

[1] https://go.chainalysis.com/2021-Crypto-Crime-Report.html

is extremely large and hard to analyze through the existing deep graph learning methods.
- **Label scarcity**. Data labels are crucial for successful applications of deep learning methods. Existing phishing scam labels (accounts) are mainly reported by different organizations or individuals, and the number of labeled data is minimal compared to the total number of accounts. Furthermore, label scarcity will become more severe in those recently-generated graph parts, creating an issue of temporal labeling imbalance.

Previous works [5, 25, 50, 51] attempt to mitigate these challenges through a graph sampling mechanism, which increases the proportion of the phishing scam nodes and reduces the size of the transaction graph. However, it could also lead to serious data distribution distortion and still suffers from the issue of temporal labeling imbalance.

In recent years, self-supervised learning (SSL) has achieved success in many areas, e.g., pre-trained language models like BERT [12], GPT series [3, 33, 36], etc.) or pre-training methods of visual models (MoCo [18], SimCLR [6], etc.). SSL is particularly good at dealing with learning problems that need to process (i) a large amount of data with (ii) very few labels. In contrast to supervised learning, SSL aims to obtain supervision signals from the data itself through different pretext tasks, which can somehow alleviate the problem of poor generalization due to overfitting, and weak robustness faced with adversarial attacks [28]. For applying SSL in graphs, rich structural information could be both inspiration and barrier for the pretext tasks design [22].

Based on the observations above, we propose a **S**elf-supervised **I**ncr**E**mental deep **G**raph l**E**arning model (**SIEGE**) for the phishing scam detection problem. It consists of a graph self-supervised learning module and a graph incremental learning module.

**First**, we design self-supervised learning pretext tasks in both spatial and temporal perspectives to learn a more complete and high-quality node representation. Our spatial pretext task aims to ensure that the learned node embeddings can represent both the initial attributes of nodes and the contexts in which the nodes are located in the graph. For smart contracts, one of the most critical components of Ethereum, we give them special consideration to learn their different functionalities in Ethereum. In our temporal pretext task, we consider the temporal variation of the node behaviors (user behaviors) and try to predict their future behaviors with their current states and contexts located. High-quality node embeddings learned from a temporal perspective also help us to determine their probabilities to be phishing scam nodes.

**Second**, for the graph incremental learning module, instead of handling the whole graph at once, SIEGE processes the graph to ensure the size of the in-memory graph is acceptable. It splits the transaction graph into pieces of suitable size (i.e., a set of Ethereum transaction blocks) and incrementally consumes them one by one in the temporal order to learn the node representation. With this way of processing Ethereum transaction data sequentially, SIEGE could adapt to the latest changes by continuously feeding new transaction data.

To conclude, our contributions are as follows,

- We present SIEGE, a novel self-supervised learning model for phishing scam detection in Ethereum. Experiments on real-world transaction data demonstrate that SIEGE achieves significantly higher performance (6% ~19% improvement in F-1 score) compared to baseline models, in both transductive and inductive settings.
- To address the challenge of huge data volume, the original Ethereum transaction data is divided into smaller splits and transaction graphs are constructed for each split. Then an incremental learning module is proposed to learn these graph splits sequentially.
- To overcome the label scarcity challenge, we propose a self-supervised learning module that incorporates two pretext tasks. These tasks enable graph neural networks (GNNs) to effectively leverage spatial and temporal information from the transaction graph splits. The learned graph neural network model could generate high-quality node embeddings, to be used in phishing scam detection finally.

## 2 RELATED WORK

We mainly introduce two research directions related to this work: (i) Ethereum phishing scam detection; (ii) deep graph learning. For (i), we will show some background knowledge about the basics of Ethereum and the Ethereum transaction graph. Then we describe the methodology of existing works for phishing scam detection on Ethereum. We will also compare the difference between Ethereum and other blockchain platforms (especially Bitcoin). For (ii), we will briefly introduce the development of GNN methods.

### 2.1 Ethereum Phishing Scam Detection

*2.1.1 Ethereum basics.* Inspired by Bitcoin [32], Ethereum [49] aims to become a next-generation blockchain platform that supports both cryptocurrency and decentralized applications (DApps). The key is the designed smart contracts on Ethereum, which can be easily deployed in a transaction and execute preset functions when called by another account. It provides convenience for users because it allows for trusted transactions between users without a third party. Specifically, in Ethereum, two types of accounts existed: external accounts controlled by users and contract accounts with code stored together. We can get a transaction graph if we abstract the account as a node and the transaction as an edge.

The approach in [7] firstly makes a systematic investigation of Ethereum via graph analysis. This work studies the characteristics (degree distribution, clustering coefficients, etc.) of transaction graphs on Ethereum and proposes a rule-based anomaly detection method. Specifically, it aims to detect accounts that have created massive abnormal smart contracts and consumed a lot of resources. And the proposed method mainly leverages expert knowledge to choose some thresholds for account behavior features and detect anomalies according to predefined rules. This method is simple and useful. The results provide some insights into the behavior pattern of accounts in Ethereum. However, it could only detect some relatively simple abnormal behaviors and requires extensive tuning efforts. For phishing scams with complex behavior patterns, it is difficult to apply in practice.

When comparing Ethereum and Bitcoin, it should be noted that there is no concept of "accounts" or "balance" for Bitcoin. The basic block of Bitcoin is the unspent transaction output (UTXO), and the transaction in Bitcoin can have multiple outputs and inputs, while the transaction in Ethereum is 1-on-1. This is part of the reason why we studied the Ethereum transaction graph. Although it is not feasible to use SIEGE directly for Bitcoin phishing scam detection, we describe two possible solutions here, which may be our future research directions. First, if we change our goal to phishing scam transaction detection, some works like [48] build the Bitcoin transaction graph with the transaction as a node and the Bitcoin flow as an edge, then our model will become applicable. Second, works like [14, 30, 31, 38] employ address clustering and de-anonymization techniques to transform the Bitcoin transaction graph into a user graph, similar to the Ethereum transaction graph. Our model can also work with such a user graph.

*2.1.2 Phishing scams in Ethereum.* The thriving Ethereum market has also bred a lot of criminal activities like phishing scam [5, 8, 20, 25, 50], Ponzi scheme [1, 9], ransomware [10], etc. There are two main categories of existing methods for the phishing scam detection problem on Ethereum. The former mainly employ shallow models such as (i) traditional machine learning methods with dedicated feature engineering [8], and (ii) some random-walk-based network embedding methods [50], such as DeepWalk [35], Node2Vec [16], etc. The latter applies some deep graph models like the graph convolutional network [24] (GCN) based methods. These methods show pretty good performance in their data collected. However, two problems limit the practical application of these methods. (i) Most existing methods are transductive, making predictions in a single fixed graph. Therefore, they do not naturally generalize to unseen nodes or sub-graphs, and for a high-throughput active blockchain platform like Ethereum, an inductive method to process fast-evolving in-coming transaction data is in need. (ii) A biased sampling process is employed in graph data generation. To alleviate the label scarcity problem and scalability problem, existing methods [5, 25] generate the graph data by sampling nodes and edges related to labeled positive data. For example, these works make a 2-hop BFS search with labeled phishing scam nodes as the start and then choose the largest weakly connected component as training graph data, which will result in different node distributions between the sampled graph and the original graph. Hence, this paper proposes SIEGE to learn useful node representations from the original unsampled transaction graph. It is also inductive and can be used for phishing node detection in an entirely new transaction graph.

Another issue is how these phishing scam detection results could be actually employed in real-world scenarios to prevent more phishing scams. We explain it from two aspects: (1) After phishing scam accounts are detected, the cryptocurrencies involved in crimes could be confiscated by the government or other orderly organizations; (2) Since Ethereum transaction records are public, users can see all the transaction records. Although the phishing scam accounts are able to transfer their cryptocurrencies out, we could send alerts to corresponding users or release all phishing scam information on a website. In this way, we could also leverage the phishing scam detection results.

## 2.2 Deep Graph Learning

Deep learning has been a great success in computer vision, speech recognition, and natural language processing due to its powerful representation learning capability. Recently, increasing attention has been paid to how to apply deep learning for non-euclidean data like graphs, i.e., graph neural networks, which is widely used in various tasks, such as social network [13], protein interface prediction [15], knowledge graph embedding [46], etc. Graph neural networks are rapidly improving and could be mainly divided into spectral and spatial methods. Spectral methods [4, 24] are based on the spectral representation of a graph. And spatial methods [17, 44] are based on the information aggregation and transformation from local neighborhood nodes.

Most GNNs could incorporate the information from node attributes and graph topology. Then the node embedding generated can be used for downstream tasks such as node classification, link prediction, graph classification (an extra readout module in need), etc.

In addition, it should be clarified that existing dynamic graph neural networks like [11, 34] cannot be employed in this problem. The key to these models is to learn the temporal relationships in a graph series by a temporal encoder (like LSTM [19]). However, these models are trained in a sequential approach due to the usage of an RNN-like structure. This will result in all trainable parameters in each time step needing to be saved until the end of back-propagation. This is unacceptable due to the huge data volume in this problem. Besides, most of them are supervised methods and are not suitable when the label scarcity problem is severe.

# 3 METHODOLOGY

In this section, we first briefly describe the Ethereum transaction graph data and formalize our problem. Then, we explain how the self-supervised learning module works, divided into two parts: spatial pretext task and temporal pretext task. In the last, we introduce our graph incremental learning module and make a summary of SIEGE.

## 3.1 Data Description

In Ethereum, all activities are completed in the form of transactions. The two parties involved in a transaction are called accounts. There are mainly two kinds of accounts in Ethereum: (i) externally owned accounts (EOA) and (ii) smart contracts. Their main difference is that smart contracts usually contain executable codes deployed by users for certain purposes [7]. Then, the Ethereum transaction can be easily formulated into a transaction graph, whose nodes are accounts and edges denote transactions.

The transaction graph is extremely large due to the huge user number and active Ethereum transactions. We are not able to directly apply graph learning algorithms on the entire transaction graph to perform intelligent detection from the perspective of computational overhead. It is also why we need an efficient training method to handle the large graph. The proposed method, detailed later, will split the graph in temporal order and utilizes an incremental graph learning strategy.

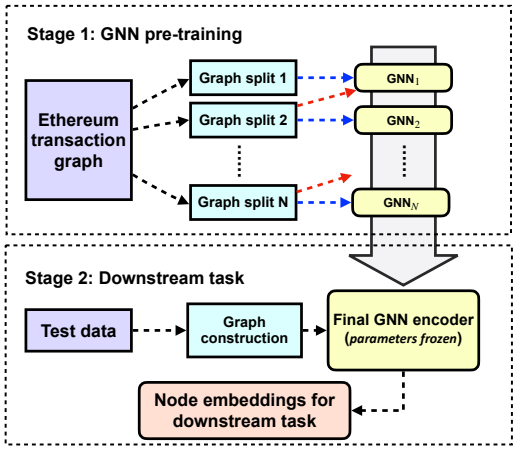


**Figure 1: The overall workflow of SIEGE, where *blue* dash lines stand for the spatial pretext task, and *red* ones stand for the temporal pretext task. For GNN models from $\text{GNN}_1$ to $\text{GNN}_N$ in the pre-training stage, the parameters of the previous encoder will be used to initialize the next one. The downstream task denotes the phishing scam nodes classification task.**

For the phishing scam detection task, in such an Ethereum transaction graph, the data labels available are almost negligible compared to the total data volume, as mentioned in Section 1. This is why we look to SSL technologies for inspiration in the design of SIEGE.

## 3.2 Problem Formulation

We first divide the original graph data $\mathcal{G}$ into $N$ splits $\{\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_N\}$ in the order of block number (same as the temporal order). For graph split $i$, $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i, \boldsymbol{X}_i)$, where $\mathcal{V}_i = \{v_1, v_2, \cdots, v_{N_i}\}$ is the set of $N_i$ nodes, $\mathcal{E}_i$ stands for the edge set and $\boldsymbol{X}_i \in \mathbb{R}^{|\mathcal{V}_i| \times d}$ is the node feature matrix. Moreover, we use $\boldsymbol{A}_i$ as the adjacency matrix of graph split $i$, where $\boldsymbol{A}_{i,m,n} = 1$ if there is an edge between node $v_{i,m}$ and $v_{i,n}$ else $\boldsymbol{A}_{i,m,n} = 0$. To conclude, each graph split is a directed graph with node attributes.

In our self-supervised learning module, the encoder GNN $g_i$, could incorporate both the information of graph topology $\boldsymbol{A}_i$ and node attributes $\boldsymbol{X}_i$, then get the node embeddings $\boldsymbol{Z}_i$. Our objective is to minimize the pretext tasks designed and get the final optimized GNN encoder $g_{\text{final}}$. Then we freeze the parameters of $g_{\text{final}}$ and employ $g_{\text{final}}$ in the test graph data to obtain a representation of the nodes used for the downstream classification task, as illustrated in Figure 1.

## 3.3 Self-supervised Learning Module

The core of SSL is obtaining the supervision signal and learning from the data itself. And the pretext task design determines the direction and objective of SSL, which is the most important part of an SSL framework. The design of pretext tasks is mainly developed from two aspects: generative and contrastive methods [29]. For a graph split aforementioned, a general pretext task could be

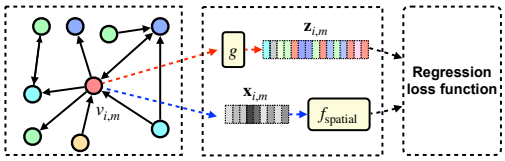


**Figure 2: An overview of the spatial pretext task in SIEGE, where $g$ is the GNN encoder, and $f_{\text{spatial}}$ is a linear layer. For the masked node $v_{i,m}$, $\boldsymbol{x}_{i,m}$ and $\boldsymbol{z}_{i,m}$ denote the initial attributes and the node embedding through $g$, respectively.**

formulated as follows,

$$\min_{g} \mathcal{L}_{\text{pretext}}(\boldsymbol{A}, \boldsymbol{X}, g) = \sum_{v_i \in \mathcal{V}_{\text{pretext}}} D(g(\mathcal{G})_{v_i}, y_{\text{pretext}_i}) \quad (1)$$

where $g$ is the GNN encoder (feature extractor), and the $y_{\text{pretext}_i}$ stands for the ground truth of node $v_i$ acquired by the pretext task. $\mathcal{G}$ is one of the graph splits. In addition, $\mathcal{V}_{\text{pretext}}$ is the node set used in the pretext task, and the discriminator $\mathcal{D}$ is used to measure the relationship between the node embedding of $v_i$ and $y_{\text{pretext}_i}$. It should be noted that our self-supervised model involves only node-level pretext tasks. As for edge-, sub-graph- or graph-level pretext tasks, we leave them for future work. After pre-training with the pretext task, the parameters of GNN encoder $g$ will be frozen to generate the node embeddings $\boldsymbol{Z} = \{z_1, z_2, \cdots, z_{N_{\text{test}}}\}$ for an input test graph $\mathcal{G}_{\text{test}}$, which will be fed into the final classifier for the downstream task.

We call this case a transductive setting, i.e., $\mathcal{G}_{\text{test}}$ and $\mathcal{G}$ are the same. If they are different, we call the case an inductive setting. Their main difference is that in an inductive scenario, the learned GNN model will be employed in a new graph.

For the phishing scam detection problem in this paper, we conclude three prerequisites for a well-designed pretext task as follows,

(1) the extracted data labels should reflect the characteristics of the data itself;
(2) it should be possible for users to obtain data labels with relatively low time complexity. Otherwise, due to the huge volume of Ethereum transaction data, the time cost could be unacceptable;
(3) domain knowledge is an important and valuable inspiration for pretext tasks. But it is also important to consider generalization issues and avoid overuse of domain knowledge when designing pretext tasks, which is likely to overfit the model.

Then, we detail the design of our spatial and temporal pretext tasks.

*3.3.1 Spatial pretext task.* For the nodes in the Ethereum transaction graph, we hope that the learned node embeddings could incorporate both their initial attributes and contexts in the graph (i.e., information on surrounding nodes and structures). To efficiently capture this kind of spatial information, inspired by [21], we design a spatial pretext task, which is illustrated in Figure 2. First, for a certain graph split $\mathcal{G}_i$, we first mask some node initial attributes in $\mathcal{G}_i$ by setting the corresponding attributes to zero. Then we calculate its node embedding $\boldsymbol{Z}_i = \{z_{i,1}, z_{i,2}, \cdots, z_{i,N_i}\}$ through GNN model $g$. Finally, for a masked node $v_{i,m}$, we employ

its learned node embedding $\boldsymbol{z}_{i,m}$ to predict its initial attributes $\boldsymbol{x}_{i,m}$. We formalize this procedure as below:

$$\mathcal{L}_{\text{spatial}}(\boldsymbol{A}_i, \boldsymbol{X}_i, g) = \frac{1}{|\mathcal{V}_i^{\text{masked}}|} \sum_{v_{i,m} \in \mathcal{V}_i^{\text{masked}}} \|\boldsymbol{z}_{i,m} - f_{\text{spatial}}(\boldsymbol{x}_{i,m})\|^2 \quad (2)$$

, where $\mathcal{V}_i^{\text{masked}}$ denotes the masked node set in $\mathcal{G}_i$. $f_{\text{spatial}}$ is the linear transformation layer. $\boldsymbol{z}_{i,m}$ and $\boldsymbol{x}_{i,m}$ are the node embedding and the initial attributes of $v_{i,m}$, respectively.

As reported in Section 1, in addition to the basic transfer functionality, smart contracts can be deployed with some executable codes for different purposes, which is the core characteristic of Ethereum. Various applications of smart contracts, such as issuing tokens, auctions, lotteries, gambling, games, etc., have been common on Ethereum. Moreover, some phishing scam nodes are also smart contracts. Therefore, it is important and helpful to leverage the information from smart contracts to detect those phishing scam nodes. To capture different functionalities of smart contract nodes, we design another additional spatial pretext task to predict their functionalities.

Specifically in the graph split $\mathcal{G}_i$, for smart contract node set $\mathcal{V}_i^{sc}$ and their initial attributes $\boldsymbol{X}_i^{sc}$, we first perform clustering on these smart contract nodes and obtain cluster index $C_m$ for each node $v_{i,m}^{sc}$. Then we predict $C_m$ with its node embedding $\boldsymbol{z}_{m,i}^{sc}$. The procedure is formulated as

$$\mathcal{L}_{\text{spatial}}^{\text{sc}}(\boldsymbol{A}_i, \boldsymbol{X}_i, g) = \frac{1}{|\mathcal{V}_i^{sc}|} \sum_{v_{i,m}^{sc} \in \mathcal{V}_i^{sc}} l(f_{\text{spatial}}^{\text{sc}}(\boldsymbol{z}_{i,m}^{sc}), C_m) \quad (3)$$

, where $f_{\text{spatial}}^{\text{sc}}$ is a linear layer and $l$ is the cross-entropy loss function.

It should be noted that most smart contracts on Ethereum could be grouped into several categories from the perspective of deployment purpose [40], such as finance, game, wallet, social, etc. Therefore this step would not introduce much noise.

*3.3.2 Temporal pretext task.* The design of our temporal pretext task is mainly motivated by two insights:

(1) In our spatial pretext task, for a node in the Ethereum transaction graph, we have considered its initial attributes and context information from the graph. On top of that, to detect phishing scam nodes, how the behavior of this node evolves in the time dimension could also be important and valuable.
(2) As mentioned before, the whole Ethereum transaction graph is too large and growing rapidly, we are not able to train it with GNN at once. As a solution, we split the whole graph in temporal order to reduce the memory cost. However, such a division breaks the overall graph structure and is likely to cause information loss.

Therefore, we hope to design a temporal pretext task that could simultaneously (i) learn the temporal variation of node behaviors and (ii) reduce the information loss caused by the graph division.

Specifically, for two adjacent graph splits $\mathcal{G}_i$ and $\mathcal{G}_{i+1}$, since each graph split contains a large amount of transaction data, there must be a certain percentage of nodes which appear in both graph splits. This set of nodes forms a bridge between the two adjacent graph splits. We use this overlap node set $\mathcal{V}_{i,o} = \mathcal{V}_i \cap \mathcal{V}_{i+1}$ to model the relationship between the two graph splits in our temporal pretext task.

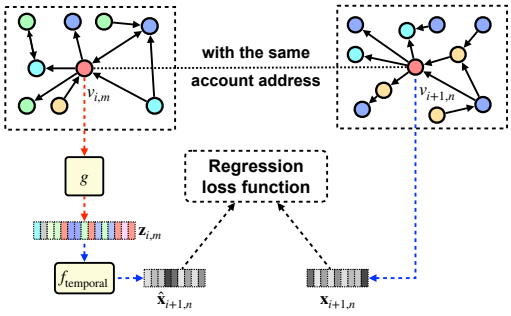


**Figure 3: An overview of the temporal pretext task in SIEGE, where $g$ is the GNN encoder, and $f_{\text{temporal}}$ is a linear layer. Two nodes $v_{i,m} \in \mathcal{V}_i$ and $v_{i+1,n} \in \mathcal{V}_{i+1}$ have the same account address in the Ethereum. $\hat{\boldsymbol{x}}_{i+1,n}$ denotes the predicted node attributes of $v_{i+1,n}$, and $\boldsymbol{z}_{i,m}$ is the node embedding of $v_{i,m}$ through $g$.**

As shown in Figure 3, for two nodes $v_{i,m}, v_{i+1,n} \in \mathcal{V}_{i,o}$ ($v_{i,m} \in \mathcal{V}_i$ and $v_{i+1,n} \in \mathcal{V}_{i+1}$) with the same account address in Ethereum, we try to make the following two vectors similar:

(1) for node $v_{i,m}$, the prediction of the future node attributes $\hat{\boldsymbol{x}}_{i+1,n}$, which is calculated from its learned node embedding $\boldsymbol{z}_{i,m}$;
(2) for node $v_{i+1,n}$ in the future, its node attributes $\boldsymbol{x}_{i+1,n}$.

Formally, we minimize $\mathcal{L}_{\text{temporal}}$ as follows,

$$\mathcal{L}_{\text{temporal}}(\boldsymbol{A}_i, \boldsymbol{X}_i, \boldsymbol{A}_{i+1}, \boldsymbol{X}_{i+1}, g) = \frac{1}{|\mathcal{V}_{i,o}|} \sum_{v_{i,m}, v_{i+1,n} \in \mathcal{V}_{i,o}} \|f_{\text{temporal}}(\boldsymbol{z}_{i,m}) - \boldsymbol{x}_{i+1,n}\|^2 \quad (4)$$

## 3.4 Graph Incremental Learning Module

Due to the huge and rapidly increasing data volume of Ethereum transactions, it is not practical to feed all transaction data into the model at once. It is why we utilize an incremental training method by feeding only partial training data at a time, which alleviates the data scalability problem.

Specifically in this module, for $v_{i,m} \in \mathcal{V}_i$ and $v_{i-1,n} \in \mathcal{V}_{i-1}$, if $v_{i-1,n}$ has the same account address with $v_{i,m}$ in the Ethereum, we will concatenate its node embedding $\boldsymbol{z}_{i-1,n}$ with $\boldsymbol{x}_{i,m}$ as the updated initial node attribute of $v_{i,m}$. Such a concatenation operation could be seen as a "connection" between adjacent graph splits, to make nodes in $\mathcal{G}_i$ aware of their state in $\mathcal{G}_{i-1}$. The reason for choosing node embedding $\boldsymbol{z}_{i-1,n}$ but not raw attributes $\boldsymbol{x}_{i-1,n}$ of $v_{i-1,n}$ is because the node embedding could contain information from its neighborhood (through GNN forward propagation), therefore provide larger receptive-field also more information to the node $v_{i,m}$.

As for GNN encoders, we employ learned $g_{i-1}$ to initialize $g_i$.

In the following, we will briefly conclude SIEGE in several steps with both the self-supervised learning module and the graph incremental learning module.

**Table 1: Initial node attributes generation process. For each node on the graph, we mainly consider three aspects of information: (1) graph structure, (2) transaction amount, and (3) transaction time. In/out transactions means transactions to/from the target nodes, respectively.**

| Type | Initial attributes |
|---|---|
| **graph structure** | in/out degree; number of in/out transactions; |
| **transaction amount** | sum/average of all/in/out transactions amount; |
| **transaction time** | span/frequency of all/in/out transactions; *mean/min/max/std/median* of all/in/out transaction intervals; |
| **others** | is smart contract or not; repeated transaction ratio of in/out transactions; |

**Table 2: Data statistics for the 5 graph splits. "Overlap" refers to the ratio of nodes in the current split which also appear in the next split. It will be used in the temporal pretext task.**

| Split | #Nodes | #Edges | Overlap | #Phish |
|---|---|---|---|---|
| 1 | 5,553,012 | 12,327,037 | 0.312 | 325 |
| 2 | 4,481,616 | 8,315,943 | 0.268 | 418 |
| 3 | 5,314,362 | 10,523,392 | 0.304 | 524 |
| 4 | 5,316,322 | 12,513,016 | 0.371 | 587 |
| 5 | 4,442,089 | 8,763,528 | - | 477 |

(1) For $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_N\}$, with each $\mathcal{G}_i = (\boldsymbol{A}_i, \boldsymbol{X}_i)$.
(2) In a graph split $\mathcal{G}_i$. Update $\boldsymbol{X}_i$ with $\boldsymbol{Z}_{i-1}$ and initialize $g_i$ with $g_{i-1}$.
(3) Obtain node embeddings $\boldsymbol{Z}_i = g_i(\boldsymbol{A}_i, \boldsymbol{X}_i)$ through the GNN encoder $g_i$.
(4) With node embeddings $\boldsymbol{Z}_i$ and GNN encoder $g_i$, minimize loss $\mathcal{L}_{\text{spatial}}$ in Equation (2), loss $\mathcal{L}^{\text{sc}}_{\text{spatial}}$ in Equation (3) and loss $\mathcal{L}_{\text{temporal}}$ in Equation (4).
(5) Obtain the final GNN encoder $g_{\text{final}}$.

It should be noted that in step (4), the spatial loss and the temporal loss are jointly optimized. And the parameters of the final GNN encoder $g_{\text{final}}$ will then be frozen to be used as a feature extractor in the downstream phishing scam classification task.

## 4 EXPERIMENTS

In this section, we evaluate the effectiveness of SIEGE on Ethereum transaction data. Firstly, we introduce our experimental setup. Secondly, we present our main experimental results with discussion and analysis, including both transductive and inductive settings. And we also conduct ablation experiments to investigate the effectiveness and necessity of each module. Then, we compare our approach with some anomaly detection methods. Finally, we make a visualization for the learned node embeddings and initial node attributes.

### 4.1 Setup

*4.1.1 Data collection and pre-processing.* We collect transactions on Ethereum that took place from January 2018 to July 2018, which is because the labeled phishing scam samples during this period are relatively more than others. To make experiments more clear, we detail our data preparation steps as follows:

(1) There are two kinds of transactions on Ethereum: (i) external transactions launched by user accounts and (ii) internal transactions launched by smart contract accounts. Since both (i) and (ii) have significant impacts on the transaction graph, we include both of them in our dataset.
(2) We then filter out those unsuccessful and zero-value money transfer transactions, which are meaningless for this task. And we get 224,377,964 transactions in total.
(3) According to block number (equivalently, timestamp), we divided the collected transaction data into 5 splits.
(4) As no available node attributes could be found in the original data for each transaction data split, we calculate the initial node attributes from several important aspects: graph structure, transaction amount, and transaction time. Then we get a directed graph with node attributes, transformed from each data split.
(5) Finally, for each graph split, we remove isolated nodes of them. Then we obtain the 5 graph splits used in our experiments.

For the node initial attributes generation process in step (4) above, we list them in Table 1. More detailed statistics of graph splits are shown in Table 2.

For phishing scam samples used for evaluation, we collect labels from the Ethereum browser[1] and some blacklists released by companies[2]. In all, we obtained 6588 unique Ethereum account IDs related to phishing scams.

*4.1.2 Hyperparameter settings and baseline models.* For the transductive setting, we train and evaluate our method in the same graph split. For the inductive setting, we train our model in $\mathcal{G}_i$ and evaluate it in $\mathcal{G}_{i+1}$ (without any training in $\mathcal{G}_{i+1}$). In each graph split, we randomly sample some nodes without phishing scam labels as normal nodes. In each graph split, we set the number of normal node labels to be three times that of phishing scam nodes. Then we conduct evaluation on these nodes with labels, train/val/test sets are randomly split with the ratio of 50%/20%/30%. It should be noted that we chose a relatively large test set ratio due to the small number of labels.

We choose a 2-layer GraphSage [17] with mean-pooling aggregator as the backbone GNN, which is inductive and contains a neighborhood sampling mechanism which could help to improve memory and computational efficiency. We employ logistic regression as our binary classifier.

In the transductive setting, we compared SIEGE with several baseline models as follows,

- Raw features described in the Table 1;
- DeepWalk algorithm [35], an unsupervised graph embedding algorithm without consideration of node attributes. It aims to make embeddings of nodes similar when these nodes have similar topological positions on the graph;

[1] https://etherscan.io/
[2] https://github.com/CryptoScamDB/blacklist

**Table 3: Prediction results in the transductive setting in terms of *precision* (P), *recall* (R), *F1 score* (F-1) and *AUC*. The results are averaged over 5 runs in all graph splits (with the standard deviation values). "no-incremental" means that we do not use incremental training.**

| Method | P | R | F-1 | AUC |
|---|---|---|---|---|
| Raw features | 0.553 ± 0.03 | 0.504 ± 0.03 | 0.527 ± 0.04 | 0.684 ± 0.04 |
| DeepWalk [35] | 0.443 ± 0.02 | 0.688 ± 0.01 | 0.539 ± 0.03 | 0.701 ± 0.02 |
| DeepWalk+features | 0.494 ± 0.02 | 0.656 ± 0.02 | 0.564 ± 0.02 | 0.717 ± 0.01 |
| GraphSage [17] | 0.563 ± 0.01 | 0.752 ± 0.02 | 0.644 ± 0.03 | 0.779 ± 0.02 |
| DGI [45] | 0.580 ± 0.02 | 0.752 ± 0.04 | 0.655 ± 0.01 | 0.786 ± 0.02 |
| **SIEGE** | **0.640** ± 0.02 | **0.824** ± 0.01 | **0.720** ± 0.01 | **0.835** ± 0.01 |
| **SIEGE**-no-incremental | 0.624 ± 0.02 | 0.808 ± 0.01 | 0.704 ± 0.01 | 0.823 ± 0.01 |

**Table 4: Prediction results in the inductive setting in terms of *precision* (P), *recall* (R), *F1 score* (F-1) and *AUC*. Since we evaluate the model in the next split (with respect to the training split), we run our experiments in the first 4 graph splits. The results are averaged over 4 runs in different graph splits (with the standard deviation values). "no-incremental" means that we do not use incremental training.**

| Method | P | R | F-1 | AUC |
|---|---|---|---|---|
| GraphSage [17] | 0.537 ± 0.00 | 0.760 ± 0.01 | 0.629 ± 0.01 | 0.771 ± 0.02 |
| DGI [45] | 0.536 ± 0.02 | 0.784 ± 0.02 | 0.636 ± 0.02 | 0.779 ± 0.03 |
| **SIEGE** | 0.610 ± 0.01 | **0.824** ± 0.00 | **0.701** ± 0.01 | **0.825** ± 0.01 |
| **SIEGE**-no-incremental | **0.614** ± 0.01 | 0.800 ± 0.01 | 0.694 ± 0.02 | 0.816 ± 0.01 |

- The combination of DeepWalk node embeddings and raw features;
- GraphSage, trained with the contrastive pretext task described in [17];
- DGI [45], a self-supervised method through maximizing the mutual information between local patch representations and high-level graph representation.

For the inductive setting evaluated in an entirely new graph, the node embeddings generated by DeepWalk will become rotated with respect to the original embedding space, as pointed out in [17], so we do not use DeepWalk baselines in this setting.

The Adam [23] is used as our optimizer. We search the learning rate from $\{0.01, 0.001, 0.0001\}$. We set the dropout ratio to 0.5. The hidden size is searched from $\{32, 64, 128, 256\}$. The weight decay rate is set to $5e^{-4}$. The mask ratio in the spatial pretext task is set to 0.15. The clustering algorithm for smart contract nodes in Section 3.3 is K-means and K is searched from $\{5, 6, 7, 8, 9, 10\}$. For mini-batch training, the batch size is set to 512. For GraphSage, we use mean-pooling as the aggregator. For DGI, following [45], we set the two-layer GCN and three-layer GraphSage-GCN as the encoders in transductive and inductive settings, respectively.

*4.1.3 Evaluation metrics.* After we get the final node embeddings, the phishing scam detection problem becomes a binary classification problem. In our experiments, we use four metrics to evaluate the model performance: (i) **Precision**, which indicates the percentage of actual phishing scam nodes among the nodes we detected. (ii) **Recall**, which denotes the proportion of our detected phishing scam nodes among all phishing scam nodes. (iii) **F-1 score**. It considers both the precision and recall scores. (iv) **AUC**, the area under the ROC curve, is typically used in binary classification tasks.

*4.1.4 Hardware of running experiments.* We run our experiments in a single machine with 4 GPUs of TITAN Xp (12GB of RAM). OS version: Ubuntu 16.04.4 LTS, and CUDA version: 11.0.

## 4.2 Results

Our results for transductive and inductive settings are shown in Table 3 and Table 4. Due to the imbalanced data, we should pay more attention to the F-1 score and AUC score. As shown in Table 3 and Table 4, SIEGE outperforms baselines by a significant margin. For the transductive setting, SIEGE exceeds baselines by about 6% ~19% in the F-1 score and about 5% ~15% in the AUC score, which strongly demonstrates the effectiveness of our model. The comparison with GraphSage and DGI, where contrastive pretext tasks are used, demonstrates the superiority of our spatial and temporal pretext tasks.

We also tried to remove the incremental learning module from SIEGE. The comparison with the original model shows that the incremental learning method is also helpful for the detection of phishing scam nodes. And SIEGE without incremental training also outperforms baselines, which demonstrates the effectiveness of our SSL modules.

For the inductive setting, both SIEGE and baselines have a slight performance drop, probably due to a slight deviation in the data distribution for evaluation. We could also find that the SIEGE outperforms baselines by more than 6% in the F-1 score and more than 4% in the AUC score, further demonstrating the effectiveness of our model when applied in inductive scenarios.

**Table 5: Ablation study of SIEGE in the transductive setting in terms of *precision* (P), *recall* (R), *F1 score* (F-1) and *AUC*. "no-temporal" means that we do not use the temporal pretext task, the same for "no-spatial".**

| Method | P | R | F-1 | AUC |
|---|---|---|---|---|
| **SIEGE** | **0.640** | **0.824** | **0.720** | **0.835** |
| **SIEGE**-no-spatial | 0.530 | 0.568 | 0.548 | 0.700 |
| **SIEGE**-no-temporal | 0.620 | 0.640 | 0.630 | 0.755 |

**Table 6: Comparison of SIEGE and anomaly detection baselines in the transductive setting in terms of *precision* (P), *recall* (R), *F1 score* (F-1) and *AUC*.**

| Method | P | R | F-1 | AUC |
|---|---|---|---|---|
| KNN [37] | 0.495 | 0.784 | 0.607 | 0.759 |
| PCA [41] | 0.425 | 0.632 | 0.508 | 0.674 |
| LOF [2] | 0.388 | 0.584 | 0.467 | 0.640 |
| One-class SVM [39] | 0.412 | 0.632 | 0.498 | 0.666 |
| Isolation Forest [27] | 0.476 | 0.712 | 0.571 | 0.726 |
| AutoEncoder | 0.422 | 0.624 | 0.503 | 0.670 |
| ECOD [26] | 0.495 | 0.800 | 0.612 | 0.765 |
| **SIEGE** | **0.640** | **0.824** | **0.720** | **0.835** |

## 4.3 Ablation Study

We performed an ablation study for the SSL module of SIEGE. As shown in Table 5, it could be found that both the spatial pretext task and temporal pretext task are important for SIEGE. Specifically, comparing the temporal pretext task (about 9% F-1 score drop after removal) and the spatial pretext task (about 17% F-1 score drop after removal), we find that the spatial pretext task could be relatively more important, which indicates that maybe the spatial relationship has a larger impact on the phishing scam detection problem.

## 4.4 Results with Anomaly Detection Baselines

Since the phishing scam detection task could also be understood as an anomaly detection task, which aims to detect phishing scam nodes as outliers.

Therefore in this section, we further compared SIEGE with anomaly detection baselines, including both classic anomaly detection methods like KNN [37], PCA [41], LOF [2], One-class SVM [39], Isolation Forest [27], AutoEncoder, and also recently proposed models like ECOD [26]. For these baselines, we employ anomaly detection library [52] as the implementation. Other experimental settings are kept the same with Section 4.1.

As shown in Table 6, SIEGE could outperform anomaly detection baselines by more than 10% in the F-1 score and more than 7% in the AUC score. And among these anomaly detection methods, the performance of ECOD is the best. These results further demonstrate the superiority and effectiveness of SIEGE.

## 4.5 Visualization

In this section, we choose the first two Ethereum transaction graph splits in Table 2 to make a feature visualization analysis, to help further understand the quality of the node embeddings learned by SIEGE. For both the two graph splits, we visualize their initial node attributes and their learned node embeddings by SIEGE at different epochs. We use t-SNE [43] as the visualization algorithm. As shown in Figure 4, for the visualization of initial node attributes (*left*), phishing/normal nodes are difficult to distinguish. After training by SIEGE with several epochs (*middle left* and *middle right*), the learned embeddings of phishing nodes and normal nodes start to aggregate. When the training of SIEGE is close to convergence (*right*), we can separate the aggregated phishing nodes and normal nodes more easily. The visualization results in this section illustrate the training process and the high quality of the learned node representations by SIEGE.

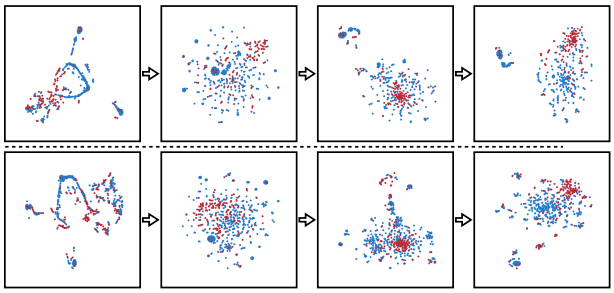

**Figure 4: Features visualization of the first two Ethereum transaction graph splits by t-SNE [43]. *Left*: node initial attributes; *Middle left*: learned node embeddings by SIEGE at 1st epoch; *Middle right*: learned node embeddings by SIEGE at 5th epoch; *Right*: learned node embeddings by SIEGE at 50th epoch. *Red* points stand for phishing scam nodes and *blue* points are normal nodes. To make the figure clear, we limit the number of points to about 400 and the ratio of normal nodes to phishing nodes is about 3:1.**

## 5 CONCLUSION

This paper aims to detect phishing scams on Ethereum. We propose SIEGE - a self-supervised deep graph learning method with an incremental training mechanism to address the label scarcity and the data scalability challenges. In our self-supervised learning module, two pretext tasks are used to extract spatial and temporal information in the Ethereum transaction graph. Our extensive experimental results on a large-scale Ethereum transaction graph dataset have shown the superiority of our method compared to baseline models. The ablation study also shows the effectiveness of the self-supervised learning module and the incremental training mechanism. SIEGE could be useful in reality for phishing scam detection to reduce financial losses and lower risks, to maintain a healthier environment for Ethereum.

## ACKNOWLEDGMENTS

This work was supported in part by NSFC under Grants 62341201, 62272224, 61872176, and 62272215, in part by the Leading Edge Technology Program of Jiangsu Natural Science Foundation under Grant BK20202001, and in part by the Science Foundation for Youths of Jiangsu Province under Grant BK20220772.

# REFERENCES


[1] Massimo Bartoletti, Salvatore Carta, Tiziana Cimoli, and Roberto Saia. 2020. Dissecting Ponzi schemes on Ethereum: identification, analysis, and impact. *Future Generation Computer Systems* 102 (2020), 259–277.

[2] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: Identifying Density-Based Local Outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data* (Dallas, Texas, USA) *(SIGMOD '00)*. Association for Computing Machinery, New York, NY, USA, 93–104. https://doi.org/10.1145/342009.335388

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

[4] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1312.6203

[5] Liang Chen, Jiaying Peng, Yang Liu, Jintang Li, Fenfang Xie, and Zibin Zheng. 2021. Phishing Scams Detection in Ethereum Transaction Network. *ACM Trans. Internet Techn.* 21, 1 (2021), 10:1–10:16. https://doi.org/10.1145/3398071

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 1597–1607. http://proceedings.mlr.press/v119/chen20j.html

[7] Ting Chen, Zihao Li, Yuxiao Zhu, Jiachi Chen, Xiapu Luo, John Chi-Shing Lui, Xiaodong Lin, and Xiaosong Zhang. 2020. Understanding Ethereum via Graph Analysis. *ACM Trans. Internet Techn.* 20, 2 (2020), 18:1–18:32. https://doi.org/10.1145/3381036

[8] Weili Chen, Xiongfeng Guo, Zhiguang Chen, Zibin Zheng, and Yutong Lu. 2020. Phishing Scam Detection on Ethereum: Towards Financial Security for Blockchain Ecosystem. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, Christian Bessiere (Ed.). ijcai.org, 4506–4512. https://doi.org/10.24963/ijcai.2020/621

[9] Weili Chen, Zibin Zheng, Jiahui Cui, Edith C. H. Ngai, Peilin Zheng, and Yuren Zhou. 2018. Detecting Ponzi Schemes on Ethereum: Towards Healthier Blockchain Technology. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, Pierre-Antoine Champin, Fabien Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis (Eds.). ACM, 1409–1418. https://doi.org/10.1145/3178876.3186046

[10] Oscar Delgado-Mohatar, José María Sierra Camara, and Eloy Anguiano. 2020. Blockchain-based semi-autonomous ransomware. *Future Gener. Comput. Syst.* 112 (2020), 589–603. https://doi.org/10.1016/j.future.2020.02.037

[11] Songgaojun Deng, Huzefa Rangwala, and Yue Ning. 2019. Learning Dynamic Context Graphs for Predicting Social Events. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis (Eds.). ACM, 1007–1016. https://doi.org/10.1145/3292500.3330919

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, 4171–4186. https://doi.org/10.18653/v1/n19-1423

[13] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia (Eds.). ACM, 417–426. https://doi.org/10.1145/3308558.3313488

[14] Michael Fleder, Michael S Kester, and Sudeep Pillai. 2015. Bitcoin transaction graph analysis. *arXiv preprint arXiv:1502.01657* (2015).

[15] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. 2017. Protein Interface Prediction using Graph Convolutional Networks. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 6530–6539. https://proceedings.neurips.cc/paper/2017/hash/f507783927f2ec2737ba40afbd17efb5-Abstract.html

[16] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi (Eds.). ACM, 855–864. https://doi.org/10.1145/2939672.2939754

[17] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 1024–1034. https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html

[18] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 9726–9735. https://doi.org/10.1109/CVPR42600.2020.00975

[19] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[20] Huawei Huang, Wei Kong, Sicong Zhou, Zibin Zheng, and Song Guo. 2021. A Survey of State-of-the-Art on Blockchains: Theories, Modelings, and Tools. *ACM Comput. Surv.* 54, 2 (2021), 44:1–44:42. https://doi.org/10.1145/3441692

[21] Wei Jin, Tyler Derr, Haochen Liu, Yiqi Wang, Suhang Wang, Zitao Liu, and Jiliang Tang. 2020. Self-supervised Learning on Graphs: Deep Insights and New Direction. *CoRR* abs/2006.10141 (2020). arXiv:2006.10141 https://arxiv.org/abs/2006.10141

[22] Longlong Jing and Yingli Tian. 2021. Self-Supervised Visual Feature Learning With Deep Neural Networks: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* 43, 11 (2021), 4037–4058. https://doi.org/10.1109/TPAMI.2020.2992393

[23] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

[24] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=SJU4ayYgl

[25] Sijia Li, Gaopeng Gou, Chang Liu, Chengshang Hou, Zhenzhen Li, and Gang Xiong. 2022. TTAGN: Temporal Transaction Aggregation Graph Network for Ethereum Phishing Scams Detection. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, Frédérique Laforest, Raphaël Troncy, Elena Simperl, Deepak Agarwal, Aristides Gionis, Ivan Herman, and Lionel Médini (Eds.). ACM, 661–669. https://doi.org/10.1145/3485447.3512226

[26] Zheng Li, Yue Zhao, Xiyang Hu, Nicola Botta, Cezar Ionescu, and George Chen. 2022. Ecod: Unsupervised outlier detection using empirical cumulative distribution functions. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[27] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*. 413–422. https://doi.org/10.1109/ICDM.2008.17

[28] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Zhaoyu Wang, Li Mian, Jing Zhang, and Jie Tang. 2020. Self-supervised Learning: Generative or Contrastive. *CoRR* abs/2006.08218 (2020). arXiv:2006.08218 https://arxiv.org/abs/2006.08218

[29] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Zhaoyu Wang, Li Mian, Jing Zhang, and Jie Tang. 2020. Self-supervised Learning: Generative or Contrastive. *CoRR* abs/2006.08218 (2020). arXiv:2006.08218 https://arxiv.org/abs/2006.08218

[30] Damiano Di Francesco Maesa, Andrea Marino, and Laura Ricci. 2016. Uncovering the bitcoin blockchain: an analysis of the full users graph. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 537–546.

[31] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. 2016. A fistful of Bitcoins: characterizing payments among men with no names. *Commun. ACM* 59, 4 (2016), 86–93. https://doi.org/10.1145/2896384

[32] Satoshi Nakamoto. 2019. *Bitcoin: A peer-to-peer electronic cash system.* Technical Report. Manubot.

[33] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* 35 (2022), 27730–27744.

[34] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. 2020. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 5363–5370.

https://ojs.aaai.org/index.php/AAAI/article/view/5984

[35] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani (Eds.). ACM, 701–710. https://doi.org/10.1145/2623330.2623732

[36] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[37] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient Algorithms for Mining Outliers from Large Data Sets. *SIGMOD Rec.* 29, 2 (may 2000), 427–438. https://doi.org/10.1145/335191.335437

[38] Dorit Ron and Adi Shamir. 2013. Quantitative analysis of the full bitcoin transaction graph. In *Financial Cryptography and Data Security: 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers 17*. Springer, 6–24.

[39] Bernhard Schölkopf, John C. Platt, John C. Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. 2001. Estimating the Support of a High-Dimensional Distribution. *Neural Comput.* 13, 7 (jul 2001), 1443–1471. https://doi.org/10.1162/089976601750264965

[40] Chaochen Shi, Yong Xiang, Jiangshan Yu, Longxiang Gao, Keshav Sood, and Robin Ram Mohan Doss. 2022. A Bytecode-based Approach for Smart Contract Classification. In *IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2022, Honolulu, HI, USA, March 15-18, 2022*. IEEE, 1046–1054. https://doi.org/10.1109/SANER53432.2022.00122

[41] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. 2003. *A novel anomaly detection scheme based on principal component classifier*. Technical Report. Miami Univ Coral Gables Fl Dept of Electrical and Computer Engineering.

[42] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, Aldo Gangemi, Stefano Leonardi, and Alessandro Panconesi (Eds.). ACM, 1067–1077. https://doi.org/10.1145/2736277.2741093

[43] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[44] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph Attention Networks. *CoRR* abs/1710.10903 (2017). arXiv:1710.10903 http://arxiv.org/abs/1710.10903

[45] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2018. Deep Graph Infomax. *CoRR* abs/1809.10341 (2018). arXiv:1809.10341 http://arxiv.org/abs/1809.10341

[46] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. 2019. Knowledge-aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis (Eds.). ACM, 968–977. https://doi.org/10.1145/3292500.3330836

[47] Jinhuan Wang, Pengtao Chen, Shanqing Yu, and Qi Xuan. 2021. TSGN: Transaction Subgraph Networks for Identifying Ethereum Phishing Accounts. *CoRR* abs/2104.08767 (2021). arXiv:2104.08767 https://arxiv.org/abs/2104.08767

[48] Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I. Weidele, Claudio Bellei, Tom Robinson, and Charles E. Leiserson. 2019. Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics. *CoRR* abs/1908.02591 (2019). arXiv:1908.02591 http://arxiv.org/abs/1908.02591

[49] Gavin Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151, 2014 (2014), 1–32.

[50] Jiajing Wu, Qi Yuan, Dan Lin, Wei You, Weili Chen, Chuan Chen, and Zibin Zheng. 2022. Who Are the Phishers? Phishing Scam Detection on Ethereum via Network Embedding. *IEEE Trans. Syst. Man Cybern. Syst.* 52, 2 (2022), 1156–1166. https://doi.org/10.1109/TSMC.2020.3016821

[51] Qi Yuan, Baoying Huang, Jie Zhang, Jiajing Wu, Haonan Zhang, and Xi Zhang. 2020. Detecting Phishing Scams on Ethereum Based on Transaction Records. In *IEEE International Symposium on Circuits and Systems, ISCAS 2020, Sevilla, Spain, October 10-21, 2020*. IEEE, 1–5. https://doi.org/10.1109/ISCAS45731.2020.9180815

[52] Yue Zhao, Zain Nasrullah, and Zheng Li. 2019. PyOD: A Python Toolbox for Scalable Outlier Detection. *Journal of Machine Learning Research* 20, 96 (2019), 1–7. http://jmlr.org/papers/v20/19-011.html