electronics

MDPI

Article

Learning to Traverse Cryptocurrency Transaction Graphs Based on Transformer Network for Phishing Scam Detection

Su-Hwan Choi and Seok-Jun Buu *

Department of Computer Science, Gyeongsang National University, Jinju-si 52828, Republic of Korea; soo3904@gnu.ac.kr

* Correspondence: sj.buu@gnu.ac.kr

Abstract: Cryptocurrencies have experienced a surge in popularity, paralleled by an increase in phishing scams exploiting their transactional networks. Therefore, detecting anomalous transactions in the complex structure of cryptocurrency transaction data and the imbalance between legitimate and fraudulent data is considered a very important task. To this end, we introduce a model specifically designed for scam detection within the Ethereum network, focusing on its capability to process long and complex transaction graphs. Our method, Deep Graph traversal based on Transformer for Scam Detection (DGTSD), employs the DeepWalk algorithm to traverse extensive graph structures and a Transformer-based classifier to analyze intricate node relationships within these graphs. The necessity for such an approach arises from the inherent complexity and vastness of Ethereum transaction data, which traditional techniques struggle to process effectively. DGTSD applies subgraph sampling to manage this complexity, targeting significant portions of the network for detailed analysis. Then, it leverages the multi-head attention mechanism of the Transformer model to effectively learn and analyze complex patterns and relationships within the Ethereum transaction graph to identify fraudulent activity more accurately. Our experiments with other models demonstrate the superiority of this model over traditional methods in performance, with an F1 score of 0.9354. By focusing on the challenging aspects of Ethereum's transaction network, such as its size and intricate connections, DGTSD presents a robust solution for identifying fraudulent activities, significantly contributing to the enhancement of blockchain security.

Keywords: graph traversing; graph neural network; DeepWalk; transformer; cryptocurrency security; fraud detection

Citation: Choi, S.-H.; Buu, S.-J. Learning to Traverse Cryptocurrency Transaction Graphs Based on Transformer Network for Phishing Scam Detection. *Electronics* **2024**, *13*, 1298. https://doi.org/10.3390/electronics13071298

Academic Editor: Zbigniew Kotulski

Received: 28 February 2024
Revised: 26 March 2024
Accepted: 28 March 2024
Published: 30 March 2024

1. Introduction

Since Bitcoin's emergence, the cryptocurrency sector has seen exponential growth, driven by its core attributes of decentralization, scalability, transparency, security, and stability. Ethereum has been at the forefront of this expansion, extending blockchain's utility beyond simple transactions to a broader range of applications through smart contracts and decentralized applications (DApps). Statistics show that, as of 2022, there are over 250,000 transactions per day for Bitcoin and 1,130,000 for Ethereum [1]. However, this rapid growth has not been without challenges. The inherent features of cryptocurrencies, particularly their anonymity and the complexity of their transaction networks, have made them a target for various cybercrimes, including investment fraud and exploitation of software vulnerabilities. A report by the Australian Competition and Consumer Commission highlighted a significant rise in cryptocurrency-related fraud, noting a 190% increase in losses between 2017 and 2018 [2].

A key challenge in combating cybercrimes lies in cryptocurrency transaction data. Ethereum's network, for instance, has accumulated approximately 2196.75 million transactions, according to Etherscan (https://etherscan.io/, accessed on 27 February 2024). As illustrated in Figure 1, visualizing just 500 nodes from Ethereum's transaction graph

shows the density and intricacy of these networks, with many interconnected nodes and edges. This graphical representation highlights how each node, representing a transaction or contract, intertwines with others to form a complex web that is both vast and intricate. This complexity makes traditional data analysis methods inadequate for processing and analyzing such vast, intertwined datasets. Governments, financial institutions, and investment advising agencies are trying to bring solutions, but the knowledge and understanding of blockchain technology and cryptocurrencies among officials and investors is still insufficient [3]. This creates a need for technology that can predict and proactively prevent fraudulent schemes on cryptocurrency networks.

In this study, we specifically chose the DeepWalk algorithm to effectively explore and understand the complex structure of the Ethereum transaction graph. DeepWalk is a useful algorithm for generating embeddings that do not consider the order of nodes in graph-structured data and works by randomly walking through each node to capture its neighbors. This approach is particularly effective at capturing deep connections and interactions between nodes, which is essential for identifying important patterns and structures in complex networks like Ethereum. By using DeepWalk, we gain important insights into each node's position within the network and its relationships with other nodes.

The output of DeepWalk is a dense vector-like embedding that reflects the key characteristics and relationships of nodes in a high-dimensional graph, enabling the numerical representation of complex relationships and patterns between nodes. This embedding is analyzed by advanced techniques such as Transformer, which, using a multi-headed self-attention mechanism, learns various node relationships and identifies complex patterns. The properties of the embeddings from DeepWalk allow the Transformer model to analyze semantic and structural node relationships more precisely. This approach outperforms traditional graph analysis methods in accurately identifying anomalous patterns, such as fraudulent behavior within the Ethereum transaction network. Therefore, we propose a novel methodology combining DeepWalk and Transformer for in-depth fraud detection in the Ethereum transaction graph, marking a significant advancement in blockchain security and showing great potential for future research and applications.

Our paper introduces a novel methodology that merges graph traversal techniques with Transformer-style neural network models by leveraging the power of DeepWalk to effectively navigate Ethereum's vast graph structure and skillfully capture the complex web of transactional interactions that define the network. This initial step is critical because it makes it possible to map out the complex and long paths and connections between transactions and contracts. Subsequently, the application of the Transformer model [4], specifically the multi-headed self-attention mechanism, facilitates a nuanced analysis of these structures. This mechanism allows the model to focus on multiple parts of the input simultaneously, highlighting the complex node relationships and transaction interactions within the Ethereum network. By computing the attention scores of different representation subspaces at different locations, the Transformer model can capture the complex dynamics and dependencies between transactions and contracts without losing sight of the overall structure. This combined approach is specifically designed to meet the unique demands of cryptocurrency transaction data analysis, providing a comprehensive and nuanced insight into the network that is vital for pinpointing fraudulent activities.

The significance of this paper lies not only in its methodological proposition but also in addressing and potentially resolving a fundamental issue in the field of cryptocurrency security. The major contributions of this research are threefold:

- A new technical approach to improve cybersecurity: The DGTSD model provides a new technical approach to enhance the security of cryptocurrency transactions and prevent fraudulent behavior. By analyzing semantic and structural node relationships in complex transaction networks, the model identifies vulnerabilities exploitable by cybercriminals. Especially, its ability to accurately detect abnormal transaction patterns in large blockchain networks like Ethereum significantly boosts the safety and reliability of blockchain technology. DGTSD introduces a new dimension to

cybersecurity research through its detailed analysis of the Ethereum transaction graph, paving the way to strengthen the security posture of blockchain-based systems.

- Deep graph traversal and classification: The DeepWalk–Transformer model, DGTSD, significantly improves the process of traversing and classifying complex and long structures in the Ethereum transaction network. By directly addressing the complexity inherent in large-scale cryptocurrency transaction data, we can simplify the task of identifying and categorizing complex patterns within vast networks. This approach marks a substantial step forward in graph-based classification methods by directly addressing the intricacies of large-scale cryptocurrency transaction data.
- Empirical validation in a real-world dataset: The DGTSD model has been rigorously tested and validated against real Ethereum transaction data, showing notable improvements in performance compared to existing methods. Its effectiveness in accurately analyzing and classifying large volumes of transactions underscores the practical utility of the model in enhancing the security of cryptocurrency networks.

The remainder of this paper is organized as follows. In Section 2, we introduce related work on fraud detection using cryptocurrency transaction data. Section 3 introduces the technical details of our proposed DGTSD. Section 4 describes the experimental evaluation of the effectiveness of the proposed approach in detecting Ethereum money laundering. Section 5 concludes this paper and presents future research directions.

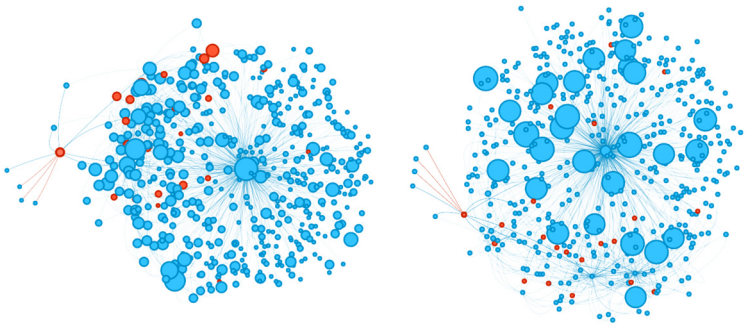

Figure 1. Ethereum transaction graph visualization. The graph depicts the complex and extensive structure of the Ethereum network, highlighting the density of interconnected nodes and edges characteristic of large-scale transaction data.

2. Related Works

In this section, we review various methodologies in the field of cryptocurrency fraud detection, ranging from traditional machine learning algorithms to advanced graph-based neural networks. Table 1 provides a detailed overview of these methodologies and categorizes them into two main types: traditional approaches and graph-based techniques.

A study focused on collecting and pre-processing Ethereum address-by-address data and then implementing the K-nearest-neighbor algorithm demonstrated the effectiveness of traditional machine learning techniques in identifying scam nodes in cryptocurrency transactions [5]. A unique approach to applying time series models to Ethereum transaction data provided insights into the temporal aspects of Ethereum transactions and demonstrated the potential of time series analysis in this area [6].

Slightly differently, a study enhanced LSTMs with an attention mechanism, improving the analysis of Ethereum transaction dynamics [7]. Another study integrated a genetic algorithm with Cuckoo Search to avoid local optima, enhancing scam node detection [8]. A study focusing on wash trades, a specific type of fraud in Ethereum, used a heuristic algorithm to detect scam nodes, highlighting the adaptability and effectiveness of traditional methodologies to address specific types of cryptocurrency fraud [9]. As detection algorithms, they proposed two approaches: circular detection using DFS (Depth-first search) and interactions between neighboring nodes.

Graph family methodologies utilize graph neural networks to capture complex node relationships. Research with temporal graph attention layers has aimed at understanding

node interactions in graphs, emphasizing the importance of temporal relationships in network data, representing a significant advance in graph-based approaches [10]. Highlighting the problems in aggregating neighbor information with different relationships, a study using reinforcement learning with GNNs reduced computational costs while maintaining transactional information. This provided an innovative direction for graph-based analysis by integrating reinforcement learning and GNNs to not only maintain transactional integrity but also reduce computational expenses [11].

As the discussion of graph-like methodologies continues, especially with advances such as integrating graph neural networks (GNNs) and attention mechanisms to capture complex relationships and temporal dynamics within network data, there is an imperative need to bridge the gap between these approaches and applying a Transformer in a traditional classification context. Our model exemplifies this intersection and underscores the need to explore aspects of the Transformer model in the context of graph-based analytics by leveraging the Transformer's power as a classifier.

As part of exploring the Transformer, understanding its attention score measurement is essential. Attention mechanisms let the model focus on various input parts, enhancing context and relevance comprehension. The prime method is scaled dot-product attention, using three inputs: Query (Q), Key (K), and Value (V), and computes similarity by calculating the dot product of Q and K, scaling it, and applying softmax to compute attention weights. These weights are applied to V for the output. Dot product scaling by the root of K's dimensionality ensures gradient stability.

Multi-head attention performs scaled dot-product attention multiple times in parallel, each using different weight matrices. This allows the model to learn information from different representations (subspaces) simultaneously. Self-attention computes the attention between elements within a sequence. Q, K and V all come from the same input. This approach is particularly useful for modeling long-range dependencies within a sequence. Cross-attention computes the attention between two different sequences. For example, it determines how much attention an element in one sequence should pay to an element in another sequence. It is often used to learn the relationship between source and target texts in tasks such as translation. Relative positional encoding [12] includes relative position information in the attention calculation, rather than absolute position between words. This allows the model to be more robust to changes in word order. These different methods are greatly improving the flexibility and scalability of the Transformer model.

There are variations on existing transformers to better handle sequence data. Among them is Transformer-XL [13], an attachment-based language model that can learn longer dependencies beyond fixed-length contexts. This model outperforms existing models by introducing a segment-level attachment mechanism and relative position embeddings. In particular, the segment-level attachment mechanism is a technique that allows the model to effectively incorporate information from previous segments into the current computation. Segment-level attachment mechanisms enable information flow between segments and improve model performance by addressing the problem of context fragmentation. This approach contributes to Transformer-XL's ability to capture a wider range of dependencies and contexts beyond the limitations of fixed-length contexts.

There are various studies that have solved the problem by utilizing only Transformer's encoder. Table 2 categorizes Transformers according to their approach and modeling method. In [14], Transformer was utilized for multiple classification of heart sound signals. Transformer is based on a self-attention mechanism, and they proposed a method for fine-tuning and classifying heart sound signals using a pre-trained Transformer encoder. The input audio data were segmented into 16 $\times$ 16 spectrogram sequences to leverage information in the time and frequency domains, and each patch was given a learnable positional embedding for spatial information recognition. In the process, the input part of the model included class tokens to clarify the classification task.

In [15], the image is represented as a sequence by dividing it into specific patch sizes, converting each patch into a token, and projecting these tokens into an embedding

dimension. In the process, classification tokens (CLS) were added to the sequence for image classification, and location embeddings were added to all tokens so that Transformer's self-attention could incorporate location information. The tokens are processed through multiple stacked Transformer encoders to extract and transform features, with the CLS tokens used to classify the image. This approach explores the potential of the Transformer model for image classification by utilizing the sequential nature of images to effectively perform classification tasks. In [16], Transformers are utilized to extract global features of an image to identify correlations between objects at a distance. CNNs were then used to extract local features of the image, i.e., local information such as the texture of objects. Through this global and local feature extraction method, they built a parallel decoupling structure between the convolutional neural network and the Transformer to improve the interaction ability. This was used to comprehensively analyze various features of the image.

Ref. [17] adopted Transformers as classifiers in multivariate time series classification problems. The embedding process is enhanced to account for the continuity of time series data, characterized by a fully connected layer and adding a nonlinear activation function, tanh. In the model structure, the multivariate nature of time series data is captured by introducing a two-tower structure, modeling the correlation between channels and between time steps for more precise learning. A gating mechanism is then adopted to effectively combine the outputs of these two towers, enabling the model to comprehensively analyze information both channel-by-channel and time-step-by-time-step.

Ref. [18] solves the binary classification problem by applying Transformer to categorical features. The model presented in this paper, TabTransformer, starts with a column embedding layer, sequentially stacks N Transformer layers, and finally generates the output through a multilayer perceptron. Each Transformer layer applies a multi-head self-attention mechanism to learn feature interactions, and a position-specific feedforward layer is used to model the nonlinear relationships between features. TabTransformer shows significant performance improvements over traditional MLP models or algorithms like GBDT and maximizes the understanding and expressiveness of deep learning models for categorical data. Given that the embeddings generated by DeepWalk have similar features to typical tabular data, it makes sense to adopt Transformer as a classifier.

Ref. [19] applied Transformer to a consumer electronics recommendation algorithm. The key distinction of this research is that it models the complex dependency relationship between users' past purchase histories through a multi-head self-attention mechanism. This allows us to understand the user's interests and behavior patterns and provide more personalized product recommendations more accurately. In the model design, we optimized the performance of the model by combining a multi-head network with an automatic gated recurrent unit (AUGRU) structure. This study is significant for its use of Transformer to solve classification problems in graph-structured data.

In [20], node embedding was performed using Transformer. First, in the process of generating node embeddings using Transformer, the target node is used as a query and the neighboring nodes are used as keys to calculate the attachment weights. Then, they generate the embedding of the target node by weighted averaging of the information from the nodes. The node embeddings obtained with Transformer contain richer semantic and topological information between nodes. They utilized Transformer to generate node embeddings, which allowed us to perform efficient network analysis.

Based on this extensive body of work, it becomes apparent that the investigation into detecting phishing nodes in Ethereum transactions is a continuing effort, utilizing a range of methodologies from traditional to graph-based algorithms. However, certain studies take a slightly different approach to detecting scam nodes or fail to reflect the extreme imbalance in the original Ethereum transaction dataset. In contrast, we have designed a model that performs reasonably well with a very small number of scam nodes while considering real-world class imbalances. Furthermore, although the proposed methods have been validated on observed data, there is limited observable data in the real-world cryptocurrency transaction environment. Therefore, it is necessary to design an effective

phishing node detection system in the real-world environment. For this purpose, it is important to develop a model that fulfills the following requirements:

- Understand and use the practical features and workings of cryptocurrency networks.
- Selectively extract only the key information needed for analysis from complex and long graph structures.
- Learn a disjoint feature space by utilizing a distance measure based on the similarity between scam nodes and legitimate nodes.

Therefore, considering the above requirements and research trends, our work proposes DGTSD, combining the advantages of DeepWalk [21] for graph embedding and Transformer for classification. This model can effectively identify scam nodes in the complex structure of the Ethereum network. DeepWalk embeds complex graph data for effective representation for training, and Transformer is a powerful tool to analyze these data and accurately classify legitimate and scam accounts.

Table 1. Related works on scam node detection in cryptocurrency transactions.

Approach	Modeling Method	Number of Scam Addresses	Proportion of Minority	Characteristics
Machine learning	K-nearest neighbors	5448	15.5%	Enough phishing address
Neural network	LSTM-FCN and BP neuralnetwork	4709	11.8%	Enough phishing address
Neural network	LSTM-attention	-	-	No specific information about dataset
Neural network	Genetic algorithm with CNN (GA-CS and CNN)	7662	78%	Too high percentage of scam address
Heuristic algorithm	Heuristic algorithm with DFS	-	-	Experiments focused on wash trades
Graph neural network	Graph attention network (GAT)	-	-	Not specifying the number of addresses in the experiment
Graph neural network	Reinforcement learning with DFS	1165	10.8%	Inject unlabeled nodes in experiments

Table 2. Works utilizing only Transformer's encoder.

Approach	Modeling Method	Characteristics
Neural network	Transformer encoder with self-attention for audio	Fine-tuning and classifying heart sound signals with spectrogram sequences and positional embeddings
	Transformer encoder with cross-attention for images	Sequential image representation with patch-based tokens, classification tokens, and location embeddings
	Convolutional Neural Network and Transformer	Global and local feature extraction with a parallel decoupling structure for crop disease classification
	Gated Transformer Networks	Multivariate time series classification with a two-tower structure and gating mechanism for channel and time-step analysis
	Transformer for categorical data	Tabular data modeling with contextual embeddings and a multi-layer perceptron for binary classification
	Transformer with multi-head self-attention for recommendations	Modeling complex dependency relationships in consumer electronics recommendation using graph embedding
Graph neural network	Transformer for node embeddings	Generating node embeddings with richer semantic and topological information for network analysis

3. Proposed Method

In this section, we take a closer look at the DGTSD architecture which is specifically designed to categorize nodes within the Ethereum Scam Transaction Network. We offer a detailed examination of our methodological framework, highlighting the novel strategy of graph sampling and the utilization of DeepWalk for profound graph traversal. Additionally, we discuss the implementation of a Transformer-based classifier, aimed at accurately

identifying and predicting scams. Figure 2 provides an overview of the proposed model. Algorithm 1 shows the pseudocode for the proposed model.

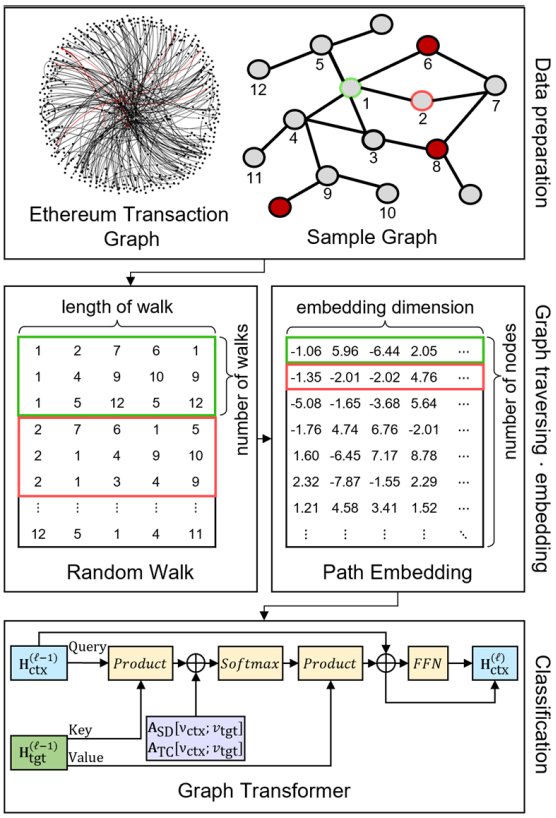

Figure 2. Overview of DGTSD. It represents the entire process from graph traversal to embedding to classification using Transformer.

3.1. Sampling Ethereum Transaction Paths

The original Ethereum transaction data consist of sizable graphs. Processing such large data is computationally inefficient and memory-limited. To address these issues, this work involves subgraph sampling, a method for extracting important parts of large graphs. We use the BFS (Breadth-First Search) algorithm [22], which starts from a selected node and systematically explores neighboring nodes:

$$S = \{v \in V | d(v, v_0) \leq k\} \tag{1}$$

where the subgraph S consists of the nodes that are within a maximum distance k from the starting node v_0 out of the entire node set V. This method makes it possible to sample any number of subgraphs from a given starting node. This methodological choice is particularly advantageous for dissecting the intricate web of transactions, enhancing our understanding of scam activities within the network.

The BFS (Breadth-First Search) algorithm enables the collection of subgraphs that contain a specified number of nodes, starting from a chosen initial node. This targeted sampling method is very efficient for managing large datasets because it selectively focuses on the most relevant segments of the graph. It has also proven to be very effective in revealing the intricacies of networks. It is also effective at capturing the inherent complexity of the network by accurately representing key patterns and relationships within the Ethereum transaction network.

3.2. Embedding Transaction Paths with Transformer Network

To analyze a graph, it is very important to first decide how to represent it. General methods include Graph Convolutional Networks (GCNs) using adjacency matrices [23]. However, in this study, we use one of the graph embedding algorithms, DeepWalk, which utilizes random walks for graph traversing. It has good processing power for large graphs and shows high performance on sparse data. This allows for better generalization of graph data.

DeepWalk performs a random walk $RW(v_i) = [v_i, v_{i1}, v_{i2}, \ldots, v_{in}]$ starting at node v_i. This allows DeepWalk to randomly traverse the graph using a random walk and then generate an embedding for each node using the Skip-Gram model. This ensures effective learning for all nodes, including those that appear infrequently in the random walk.

In particular, the Ethereum transaction graph is a digraph with directed edges. It is important to note that DeepWalk takes this into account when performing random walks: if a random walk has no neighbors, it will terminate, which means that the random walk has a "from" edge but no "to" edge from that node. This shows that DeepWalk considers the direction of the edges.

The embeddings generated by DeepWalk are fed to the Transformer classifier [4] to handle the characteristics of graph data. Figure 3 is a visualization of how Transformer plays a role in DGTSD. Transformer in the DGTSD model is designed specifically for the characteristics of graph data. Unlike traditional Transformer models that handle sequential data, this model is designed to effectively handle unordered graph data. It does not use positional encoding because there is no notion of order among the vectors in the embedding. Instead, we focus on analyzing the complex relationships between nodes and the unique characteristics of each node. In the context of natural language processing, positional encoding is a technique that allows a Transformer model to recognize the order or position of words in sequence data.

The application of the Transformer model is essential for effectively interpreting the complex nature of graph-structured data and the interactions between nodes. When analyzing graph embeddings generated by DeepWalk, Transformer's multi-head attention mechanism plays an important role in deeply understanding the complex relationships between nodes and the unique characteristics of each node. Unlike the traditional Transformer, which processes sequential data, this model is designed to account for the unordered nature of graph data and does not use positional encoding. This design focuses on the interactions between embedding vectors without any notion of order, which allows us to better analyze the complex connections between nodes and the unique characteristics of each node. These characteristics of Transformer play an important role in assessing the impact of each node on other nodes, especially in identifying potential fraud within the Ethereum transaction network, enabling a deeper understanding and analysis of complex graph-structured data.

Choosing the right attention score measurement method is important when using Transformer models to improve the learning speed and performance of the model. At this point, it is essential to consider the data coming in as input to the Transformer. DGTSD adopts a multi-head attention technique for scam node classification. This is achieved by performing multiple parallel runs of scaled dot-product attention, each with a different weight matrix, to better understand these complex connections and patterns. This allows the model to learn different aspects of relationships and information simultaneously, and as a result, each node can aggregate information from other nodes to strengthen its own embeddings. In complex data structures, such as the Ethereum transaction graph, many different types of connections and patterns can exist. By allowing these different connections and patterns to be analyzed in different representational spaces, multi-head attention techniques can capture fine-grained relationships and patterns that single-head attention techniques may miss. This allows the model to better understand the connectivity and patterns at each node, which in turn allows it to make more accurate classifications and predictions.

The multi-head attention mechanism within Transformer models the interaction between these embedding vectors to learn important characteristics and patterns within the embedded graph data. This is expressed as follows:

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^O \tag{2}$$

each head_i is calculated using scaled dot-product attention:

$$head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right) \tag{3}$$

$$W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k},\ W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k},\ W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v},\ W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$$

where d_{model} is the embedding dimension of the model, h is the number of attention heads, and d_k, d_v are the dimensions of the key and value vectors, respectively. Through this process, Transformer learns how the embedding of each node is related to the embeddings of other nodes. This allows the model to understand complex patterns within the Ethereum transaction graph and identify unusual patterns, such as phishing scams.

The attention function mentioned above is scaled dot product attention, which is defined as follows:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4}$$

Which computes the relationship between the input query (Q), key (K), and value (V) vectors. A softmax function determines the weights of the input vectors, which are used to combine the value (V) vectors.

Each layer of the Transformer then also contains Position-wise Feed-Forward Networks. These are fully connected layers that operate independently at each position:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{5}$$

where W_1, W_2, b_1, b_2 are the network parameters. This network performs operations independently at each position of the embedding vector to provide additional nonlinear transformations.

The output of each sublayer is subjected to layer normalization and residual connections. This is expressed as follows:

$$LayerNorm(x + Sublayer(x)) \tag{6}$$

where $Sublayer(x)$ is the output of each sublayer, and x is the input to the sublayer. This stabilizes the model's learning and allows it to learn effectively from deeper network structures.

Finally, the output of the Transformer model is passed through the Linear Layer and into the softmax function for determining whether a particular node is involved in a phishing scam:

$$Output = softmax(xW + b) \tag{7}$$

where W and b are the parameters of the final classification layer. Through this process, the model presented in this paper can achieve high performance in phishing fraud detection based on a deep understanding of the graph-structured Ethereum transaction data.

In Transformer, attention score computation significantly improves the accuracy of information processing by allowing the model to identify and emphasize relevant information within the input. We introduce the following attention score calculation optimized for node embedding output from DeepWalk:

- Dynamic Dimension Scaling: To account for the complexity and diversity of graph data, we dynamically adjust the dimensionality (d_k, d_v) of each attention head. This allows the model to process more diverse information and focus on specific patterns or relationships with greater granularity.

- Structural Context Awareness: By weighting each node to account for its structural position and role in the graph, richer contextual information is incorporated into the Attention Score calculation. This is especially important for effectively identifying nodes that play a significant role in the Ethereum transaction graph (e.g., hub nodes, bridge nodes) and reflecting their influence in the model.
- Pattern-Based Weight Adjustment: By adjusting weights for node pairs exhibiting certain patterns or relationships during training, we enable the model to identify unusual behaviors more accurately, such as phishing scams.

In setting up the loss function for model training, we assigned different weights to each class based on the degree of imbalance to address class imbalance. This strategy was then applied to the Focal Loss to focus training on more difficult and misclassified cases [24]. By implementing this weighting scheme, the loss contribution from minority groups is significantly enhanced. This intentional amplification allows the model to allocate more attention and sensitivity to underrepresented samples, promoting a more balanced learning process. The formula for this is as follows:

$$FL(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t) \tag{8}$$

where p_t is the probability of a particular class predicted by the model and α_t is the weight for class t. This can be set differently for each class. Assign small values to classes that occur frequently and large values to classes that occur infrequently. The gamma is typically set to a positive value, so that the larger the value, the larger the penalty for samples that the model incorrectly predicts. The larger the gamma, the more the model focuses on difficult examples.

To evaluate and optimize the model, we use precision, recall, and F1 score. The relevant formulas are as follows:

$$Precision = \frac{TP}{TP+FP} \tag{9}$$

$$Recall = \frac{TP}{TP+FN} \tag{10}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{11}$$

where TP (True Positives) are nodes that the model correctly predicted as scams when they should have been scams, and TN (True Negatives) are nodes that the model correctly predicted as not scams when they should have been legitimate. False Positives (FP) are when the model incorrectly determined a legitimate node to be a scam, and False Negatives (FN) are when the model incorrectly determined a legitimate node to be a scam.

The DGTSD methodology proposed in this work utilizes DeepWalk and Transformer networks to identify scam nodes in the Ethereum transaction graph. The effectiveness of this approach is based on the following key factors:

- Effective learning of high-dimensional graph structure: The process of embedding relationships between nodes in the graph structure using DeepWalk captures important structural patterns within complex networks. This is essential for modeling the complexity of transactions by capturing hidden connections between nodes and the overall dynamics of the graph.
- Application of Transformer networks: The Transformer model performs deep learning on these embeddings to characterize each node and identify scam nodes. The multi-head attention mechanism of the Transformer puts more weight on the relationships between important nodes to improve classification accuracy.
- Experimental validation: Through experiments on different walk lengths and node counts, we show that our methodology can effectively learn the complex characteristics of Ethereum transaction data and use them to accurately identify scam nodes.

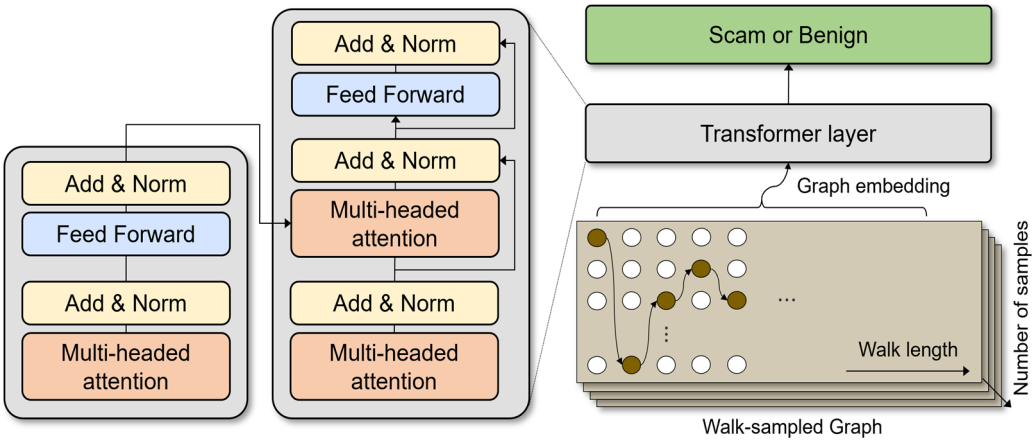

Figure 3. Detailed structure of Graph Transformer.

Algorithm 1: Deep Graph Traversal Transformer for Scam Detection (DGTSD) Training

Description: Training a DGTSD on the Ethereum Phishing Transaction Network for scam detection, leveraging deep graph traversal and Transformer-based feature extraction.
Input:
- ESTN: Ethereum Scam Transaction Network
- num_epochs: Number of training epochs
- num_classes: Number of classes (e.g., phishing/non-phishing)
- embedding_size: Dimension of node embeddings
- learning_rate: Learning rate for optimizer
- class_weights: Weights for each class to handle imbalance
- focal_gamma: Gamma parameter for Focal Loss
Output:
- dgtsd_model: Trained DGTSD model for scam detection
Initialization
1: function train_DGTSD(ESTN, num_epochs, num_classes, embedding_size, learning_rate, class_weights, focal_gamma)
2: Load ESTN and preprocess to extract node features and edges.
3: Initialize a Graph-Based Transformer with specified embedding_size and num_classes.
4: Set up the optimizer with learning_rate and FocalLoss with class_weights and focal_gamma.
Graph Traversal
5: for epoch = 1 to num_epochs do Deep Graph Traversal
6: Perform deep graph traversal using BFS to sample connected subgraphs.
7: Apply Skip-Gram on these subgraphs to generate deep contextual node embeddings.
Transformer Processing
8: Convert embeddings and labels into tensors suitable for the Transformer model.
9: Feed node embeddings through the Graph-Based Transformer. Loss Computation and Model Update
10: Calculate loss using Focal Loss, emphasizing imbalanced class representation.
11: Perform backpropagation to update the parameters of the Graph-Based Transformer.
12: Record and display training metrics such as accuracy, precision, recall, and F1 score.
Model Evaluation
13: Test the model on a separate dataset, computing and reporting evaluation metrics.
14: end for
15: return dgtsd_model

4. Experiments

4.1. Datasets and Implementation

In this study, we utilized a graphical Ethereum transaction history dataset. We followed the methodology presented in our previous work [25]. We crawl the Ethereum label cloud of phishing accounts reported by Etherscan, categorized as "fake phishing," and then leverage the API provided by Etherscan (https://etherscan.io/apis, accessed on 27 February 2024) to obtain a large transaction network with edges directed and weighted by two layers of BFS. We obtain a large graph with 2,973,489 nodes and 13,551,303 edges. A node represents an Ethereum trading account, and an edge represents a transaction history between accounts. Each edge contains transaction date and transaction volume information. If a node's label value is 0, it is a benign node, and if it is 1, it is a scam node. Out of 2,973,489 nodes,

the number of scam nodes is 1165. This indicates that the number of benign nodes is approximately 2500 times greater than that of scam nodes, highlighting a significant class imbalance.

To facilitate the experiment, we extracted a subgraph from the original Ethereum transaction data. We remove scam nodes with no edges and include the remaining 1071 nodes in the subgraph first. From each scam node, we use the BFS algorithm to explore to a certain depth and add nodes to the subgraph. If the subgraph does not contain the predetermined number of nodes after this process, we add them to the subgraph in the order of the highest degree of the nodes in the original data. After this process, the subgraph for the experiment is finally complete. Information about the subgraphs is in Table 3. Table 3 shows data points for different subgraph sizes (30,000 to 50,000 nodes), including number of edges, average degree, number of connected components, class imbalance, and density. The number of connected components refers to the total number of connected components, the set of nodes that are connected to each other within the graph. A strongly connected component is the largest set of nodes in the directed graph such that every pair of nodes can reach each other. Table 3 shows that the density of the subgraphs is relatively low. DeepWalk's use of random walks for graph traversal is well suited to sparse graphs because it captures structural information well and can effectively learn relationships between nodes with few connections.

Take the subgraph we created earlier and use DeepWalk to create an embedding for each node. The dimensionality of the embedding is fixed to 64 dimensions. Take this embedding data and label data and divide them into a train set and a test set. At this time, we first divide the scam nodes in a ratio of 7:3, and then fill the test set with as many legitimate nodes as there are scam nodes in the test set. The remaining legitimate nodes are placed in the train set. The reason for this is to prevent the F1 score from increasing as the number of nodes in training increases, simply by dividing by 7:3 for the entire dataset. This allows for a relatively objective evaluation of the model that is not affected by the number of nodes. We then utilize the Transformer classifier to perform scam node classification. To evaluate the performance of the model, we consider three classification algorithm evaluation metrics: precision, recall, and F1 score.

Information about the structure, number of parameters, and hyperparameters for the models we designed can be found in Tables 4 and 5. Table 4 shows that high performance can be extracted with relatively few parameters. Table 5 shows the values of each hyperparameter applied when training DGTSD. First, to perform DeepWalk, we perform a random walk and Skip-Gram on the Ethereum transaction graph. The num_walks and walk_length of the random walk were chosen to ensure broad coverage of the graph structure, while maintaining a balance between local and more distant neighborhood context. With a larger number of walks per node and a significant walk length, the model can capture a variety of transaction patterns, including potential phishing behaviors inherent in the transaction data. The vector_size of the Skip-Gram parameter was chosen to ensure that the dimensionality of the feature vector effectively represents the subtle relationships in the transaction graph, while not overly complicating the model. Window was chosen to optimally capture the immediate transaction context around each node.

As parameters for training the DGTSD model, class_weight and gamma play an important role. Given the imbalanced nature of phishing detection tasks, where phishing instances are much rarer than legitimate transactions, the class_weight helps to mitigate the imbalance by emphasizing the importance of accurate classification of phishing cases. The degree of imbalance in the data should be well considered while setting the values of the parameters. Focusing too much on a small number of classes will slow down learning, and setting too small a value will prevent the model from focusing on the small number of classes, which will easily lead to a poor recall value among the model's evaluation metrics. In the actual model training process, we tested several values to find the optimal value. Gamma is a parameter of Focal Loss, and like class_weight, we set an appropriately high value to focus on the minority classes. It has the same effect as cross-entropy loss when $gamma = 0$.

Table 3. Analysis of statistical characteristics by graph size of the Ethereum transaction network.

Graph Size	Number of Edges	Average Degree	The Number of Connected Components	Class Imbalance Rate (%)	Density (%)
30,000	774,193	51.62	22,600	3.7%	0.086%
40,000	994,476	49.73	30,247	2.75%	0.062%
50,000	1,389,597	55.59	36,784	2.19%	0.055%

Table 4. Number of parameters for the DGTSD models.

Layer (Type: Depth-idx)	Param #	Param #
GraphTransformer	[1, 10, 2]	--
├─ TransformerEncoder: 1-1	[1, 10, 64]	--
│ └─ ModuleList: 2-1	--	--
│ │ └─ TransformerEncoderLayer: 3-1	[1, 10, 64]	281,152
│ │ └─ TransformerEncoderLayer: 3-2	[1, 10, 64]	281,152
├─ Linear: 1-2	[1, 10, 2]	130
Total params: 562,434		

Table 5. Hyperparameters used in DGTSD training.

Type	Hyperparameter	Description	Value
Random walks	num_walks	Number of walks to be performed for each node.	80
	walk_length	Length of each random walk.	100
Skip-Gram model	vector_size	Dimensionality of the feature vectors.	64
	window	Maximum distance between the current and predicted word.	5
Transformer model	embedding_size	Size of each input token's embedding vector.	64
	num_classes	Number of classes for classification.	2
	num_layers	Number of Transformer Encoder layers.	2
	num_heads	Number of heads in the multi-head attention models.	4
	dropout_rate	Dropout rate in the Transformer Encoder.	0.1
Traininghyperparameters	batch_size	Batch size used in the DataLoader for training and testing.	64
	learning_rate	Learning rate for the Adam optimizer.	0.0005
	num_epochs	Number of epochs to train the model.	100
	class_weights	Weights for the classes in the Focal Loss.	[1, 2500]
	gamma	Gamma parameter in the Focal Loss.	3.0

4.2. Comparison DGTSD to Other Models

Comparative analysis of DGTSD models further clarifies the effectiveness of the methodology. Comparisons with LINE, node2vec, LSTM, GCN, GAT, and other algorithms show that DGTSD outperforms these traditional and state-of-the-art graph analysis techniques. DGTSD effectively identifies scam nodes by achieving excellent F1 scores despite high-dimensional data and class imbalance issues. This suggests that the DGTSD methodology is particularly effective at recognizing complex patterns in Ethereum transaction graphs and accurately classifying scam nodes within them.

Table 6 compares our DGTSD model to the following methods:

LINE [26], an algorithm designed for embedding large-scale information networks, adopts an approach that considers relationships in both one and two dimensions.

node2vec [27] is an effective method for embedding complex connections between nodes in a network, capturing structural and neighborhood information based on random walks.

LSTM [28] is a form of RNN suitable for sequential data processing and is widely used in time series data and natural language processing, with strengths in modeling long dependencies.

Table 6. Comparison of DGTSD with other models.

Model	Graph Size = 30,000			Graph Size = 40,000			Graph Size = 50,000		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
LINE [24]	0.5023	1.0000	0.6687	0.5923	0.9969	0.6680	0.5008	1.0000	0.6674
node2vec [25]	0.5000	1.0000	0.6667	0.5000	1.0000	0.6667	0.5047	0.9938	0.6694
LSTM [26]	0.6837	0.9969	0.8111	0.6772	0.9938	0.8055	0.6653	0.9938	0.7970
GCN [21]	0.5016	1.0000	0.6680	0.5016	1.0000	0.6680	0.5023	0.9969	0.6680
GAT [27]	0.5000	1.0000	0.6667	0.5000	1.0000	0.6667	0.5008	1.0000	0.6674
GNN [28]	0.6893	0.8042	0.7423	0.8140	0.7609	0.7865	0.8121	0.8089	0.7902
GCAE [29]	0.7664	0.9831	0.8614	0.7303	0.9867	0.8393	0.7158	0.9684	0.8305
Ours	**0.9067**	**0.9628**	**0.9339**	**0.8840**	**0.9907**	**0.9343**	**0.9325**	**0.9383**	**0.9354**

GCN [23] is a neural network architecture suitable for processing graph-structured data, which is useful for understanding the relationships between nodes and integrating information from neighboring nodes.

GAT [29] is a model that applies an attention mechanism to graph-structured data to better understand important relationships between nodes.

GNN [30] is a model that applies a network embedding algorithm to analyze the transaction network constructed based on data collected using web crawlers, and then applies GCN to perform classification based on the extracted features.

GCAE [31] applies the idea of autoencoder to graph data. An autoencoder is a neural network structure that compresses input data into a low-dimensional representation (encodes) and then restores them back to their original dimensionality (decodes). GCAEs apply this concept to graph data, allowing them to efficiently encode and decode nodes or entire structures in a graph. GCAEs can be utilized to solve the problem of detecting scam nodes by looking at it as an outlier detection problem.

As depicted in Table 6, it is evident that the DGTSD model surpasses other models in terms of F1 score. Graph embedding algorithms like LINE and Node2Vec had very low performance regardless of the number of nodes. GNN-like models such as GCN and GAT also did not perform well. LSTM outperformed the other models, indicating that LSTM can interpret and remember sequential information inside the graph.

A model with an F1 score close to 0.6667 means that it is close to 50% accurate. There are several reasons for this metric. First, the degree of class imbalance is very high, so we apply Focal Loss to initially train the model by focusing on scam nodes. As training progresses, the model gradually learns about the legitimate nodes. At this point, high-performing models learn the information about the legitimate nodes relatively well while retaining the information about the scam nodes, whereas the inferior models lose the generalized information about both legitimate and scam nodes, resulting in an accuracy of less than 50% on the test set.

The prioritization of the F1 score as a key performance indicator stems from our meticulous approach to test set construction, as outlined in Section 4.1. We ensured an equal representation of legitimate and scam nodes across all testing scenarios, thereby establishing a balanced and fair benchmark for model evaluation. This preparation underscores our commitment to accuracy and reliability in model comparison, further highlighting the DGTSD model's exceptional performance in distinguishing between legitimate and scam nodes within the network.

4.3. Performance over Length and Number of Nodes

In our investigation, the adaptability of the DGTSD model to varying lengths of walks was assessed. To ensure consistent experimental conditions, the same subset of subgraphs was used across all tests. This controlled approach allowed for a direct comparison of the impact of different walk lengths on the model's performance. Detailed in Table 7, we explored a spectrum of walk lengths ranging from 40 to 100. The data obtained from these experiments revealed that a walk length of 100 provided the most favorable results for

this dataset, suggesting an enhanced capability of the model to capture and learn from the underlying patterns at this scale.

Table 7. Performance as a function of walk length and number of nodes.

Walk Length	Graph Size = 30,000			Graph Size = 40,000			Graph Size = 50,000		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
40	0.8560	0.9752	0.9117	0.8917	0.9690	0.9288	0.8837	0.9383	0.9102
60	0.8739	0.9443	0.9077	0.8753	0.9783	0.9240	0.8776	0.9290	0.9025
80	0.8342	**0.9659**	0.8953	**0.8980**	0.9536	0.9249	0.8667	0.9228	0.8939
100	**0.9067**	0.9628	**0.9339**	0.8840	**0.9907**	**0.9343**	**0.9325**	**0.9383**	**0.9354**

However, it is important to note that an increase in walk length did not uniformly lead to better learning outcomes. For instance, within a graph comprising 40,000 nodes, a considerably shorter walk length of 40 emerged as the second most effective configuration. This nuanced finding emphasizes the importance of a tailored approach, advocating for thorough experimentation with a variety of walk lengths to ascertain the most effective one for each unique dataset. This strategy is essential for optimizing the model's performance, considering the specific characteristics and complexities of the data in question.

4.4. Ablation Study

In our ablation study, we conducted two sets of experiments to evaluate the effectiveness of our method. First, we replaced DeepWalk with other algorithms to assess whether it stands as the optimal graph embedding algorithm. Conversely, we substituted Transformer with alternative classifiers to ascertain whether Transformer indeed represents the best classifier for our purposes. Table 8 illustrates that, in place of DeepWalk, we utilized LINE, SDNE, and Walklets for comparison. SDNE [32] employs an autoencoder to produce non-linear embeddings, adeptly capturing the intricate high-dimensional relationships between nodes. Walklets [33] is an algorithm that varies the length of random walks to include multiscale structural information and more accurately reflect a range of node relationships. This allows them to better reflect a wide range of relationships between nodes. For the classifier comparison, we experimented with RandomForest [34], SVM [35], and CNN [36] as alternatives to Transformer. The evaluation of F1 scores reveals that the DGTSD model exhibits the most robust performance.

Table 8. Ablation study without DeepWalk or without Transformer.

Model	Graph Size = 30,000			Graph Size = 40,000			Graph Size = 50,000		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Graph traversing									
LINE + Transformer	0.5008	1.0000	0.6674	0.5047	0.9876	0.6681	0.5008	1.0000	0.6674
SDNE + Transformer	0.7875	0.9752	0.8714	0.8378	0.8793	0.8580	0.7689	0.9753	0.8599
Walklets + Transformer	0.5000	1.0000	0.6667	0.5000	1.0000	0.6667	0.5000	1.0000	0.6667
Path Modeling									
DeepWalk + RandomForest	0.9610	0.4582	0.6205	0.9793	0.4396	0.6068	0.9739	0.3456	0.5103
DeepWalk + SVM	0.8494	0.8483	0.8482	0.8567	0.8529	0.8526	0.8063	0.8056	0.8054
DeepWalk + CNN	0.7431	0.9226	0.8232	0.7186	0.9867	0.8207	0.6465	0.9877	0.7814
DeepWalk + Transformer	**0.9067**	**0.9628**	**0.9339**	**0.8840**	**0.9907**	**0.9343**	**0.9325**	**0.9383**	**0.9354**

4.5. Case Analysis

In the case analysis, our aim was to decode the DGTSD model's functionality and assess its node classification accuracy. Analyzing 20 iterations of the DGTSD model as presented in Table 9, and leveraging data sourced from etherscan.io, we scrutinized the model's binary classification efficacy across a spectrum of node types. Figure 4 is a visual-

ization of each case. It was observed that a significant portion of the classifications were predominantly aligned with the "low transaction wallets" category. This concentration suggests that the DGTSD model is guided by complex, model-specific rules, whose subtleties may not be instantly clear. This observation highlights the model's deep analytical prowess and indicates an area ripe for further investigation to unravel the mechanisms behind these classifications.

Table 9. A table that summarizes which types of nodes are present in each case. Node address contains only one representative node.

Case	Type	Visualization	Node Address
Case 1 (Benign predicted to be Benign)	**Fraudulent wallets** Wallets bought and sold by multiple users Wallets with few transactions	(a)	0x2c974b2d0ba1716e64 4c1fc59982a89ddd2ff724
Case 2 (Benign predicted to be a Scam)	A wallet that was scammed and sold all at once Wallets sold to multiple users **Wallets with few transactions**	(b)	0xfe1098e7ade721eb7b0 6fa5f766097e52a7c16e6
Case 3 (Scam predicted to be a Scam)	Wallets with a history of transactions with scam nodes Wallets bought and sold with multiple users **Wallets with a lot of 0 ETH transactions** Wallets with few transactions	(c)	0x903bb9cd3a276d8f18f a6efed49b9bc52ccf06e5
Case 4 (Scam predicted to be Benign)	Wallets with similar patterns to other benign node nodes Wallets that have transacted with many different wallets **Wallets with few transactions**	(d)	0x2b1c266e2a39a7a30ae d472fad47d6b6ab953f7e

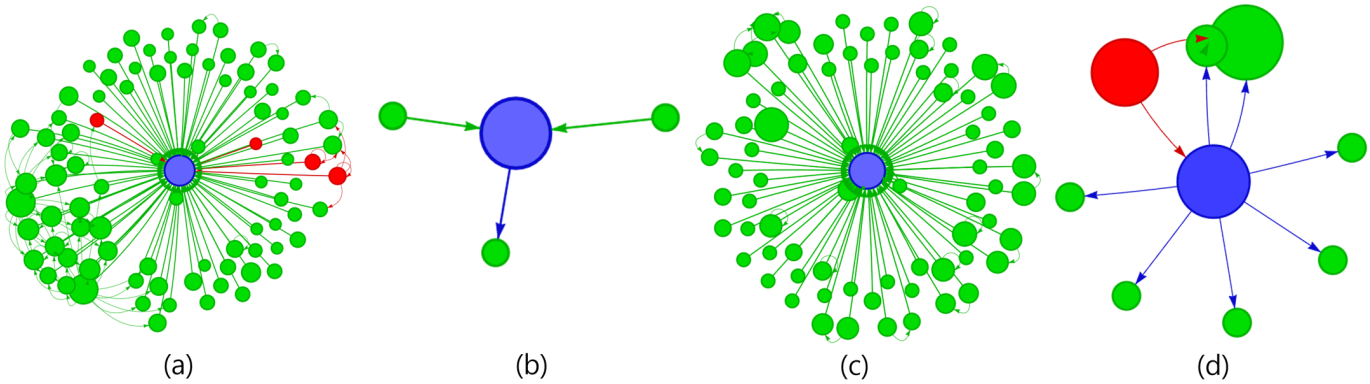

Figure 4. Visualization of representative nodes for each of the four cases in case analysis.

Among the cases that predict well as legitimate nodes when they are legitimate nodes, the most prominent is the "fraudulent wallet (a)" case. In this case, it is likely that the transactions were made purely for investment purposes. Conversely, there are many cases wherein DGTSD primarily predicts a scam node when it is a legitimate node, but most of them are "wallets with low transaction history (b)". Among the cases that are predicted to be scam nodes when they are scam nodes, one case that stands out is "wallets with an overwhelming amount of 0 ETH transactions (c)". We recently started to recognize the existence of fraudulent schemes utilizing 0 ETH transactions. In terms of the actual scam nodes predicted as benign, there are not many special cases, and most of them are "wallets with low transaction history (d)".

5. Conclusions

In this paper, we present the Deep Graph Traversal based on Transformer for Scam Detection (DGTSD), a novel method for detecting cryptocurrency crime. DGTSD utilizes DeepWalk and Transformer for graph exploration to effectively classify scam nodes in the Ethereum transaction graph and has proven to be superior to other methods in our

experiments. We also implemented advanced attention scoring methods such as dynamic dimension scaling, structural context recognition, and pattern-based weighting to better learn and identify important features and patterns in complex data structures like the Ethereum transaction graph. Although we succeed in identifying both legitimate and scam nodes with limited transaction records, the precise basis of our accuracy remains unclear.

The uniqueness of the DGTSD methodology represents a significant advance in transaction pattern analysis and fraud detection, especially within the Ethereum network. While existing methods rely on simple statistical approaches or limited graph characteristics, DGTSD goes beyond this with deep graph traversal and a Transformer-based classifier that directly interprets the complexity of the data. The methodology aims to uncover hidden patterns in transactions and predict fraudulent transactions based on them. It includes the following key technical innovations:

- High-dimensional graph structure learning: DGTSD considers the complex connectivity structure of the Ethereum network to capture the relationships between nodes. The combination of node embeddings using DeepWalk and Transformer networks captures the structural patterns and hidden connections that are very important within complex networks.
- Pattern recognition in complex data structures: DGTSD deepens the characterization of graph data through a multi-head attention mechanism. This plays an important role in assessing the impact of each node on other nodes and identifying potential fraud.
- Experimental validation and practical applicability: The methodology has been validated through extensive experiments, showing that it is applicable to real Ethereum network data. This is an important step in bridging the gap between theoretical models and real-world applications.

Future research will focus on the following factors:

- Extending to real-world applications: This study focused primarily on theoretical aspects; future research needs to validate the effectiveness of these methodologies for application to real-world problems.
- Gas fee analysis: Our research will focus on analyzing gas fees incurred in Ethereum transactions as a potential indicator of anomalous transactions [37]. Gas fees are the cost of executing transactions on the Ethereum network and may indicate unusual patterns in fraudulent transactions. This is expected to further improve the accuracy and reliability of our fraud detection model.
- Expanding the fraud node database: We plan to expand the fraud node database currently in use to cover different types of fraud cases. This will allow the model to recognize and classify a wider range of fraudulent behavior.
- Addressing class imbalance: To address the issue of class imbalance between fraudulent and legitimate nodes, we plan to perform a detailed statistical analysis of the characteristics of fraudulent nodes. This will help the model to identify and classify fraudulent nodes more accurately.

DGTSD provides a new perspective on cryptocurrency crime, specifically fraud detection in the Ethereum transaction graph. The method outperforms existing fraud detection methods and makes an important contribution to effectively handle graph-structured data. However, there are still many issues and room for improvement, which will be the focus of future research. Research directions such as analyzing gas fees, expanding the scam node database, and addressing class imbalance will be important steps to further develop the DGTSD methodology and contribute to the field of cryptocurrency fraud detection. These research directions are essential to develop more sophisticated and efficient fraud detection systems, which will contribute to ensuring the safety of cryptocurrency transactions and gaining the trust of users.

Author Contributions: Conceptualization, S.-J.B.; Formal analysis, S.-H.C.; Funding acquisition, S.-J.B.; Investigation, S.-J.B.; Methodology, S.-J.B. and S.-H.C.; Visualization, S.-H.C.; Writing—review and editing, S.-J.B. and S.-H.C. All authors have read and agreed to the published version of the manuscript.

Funding: Following are results of a study on the "Leaders in Industry-university Cooperation 3.0" Project, supported by the Ministry of Education and National Research Foundation of Korea.

Data Availability Statement: The data presented in this study are openly available in Xblock datasets at http://xblock.pro/#/search?types=datasets (accessed on 28 February 2024). [25].

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Coinmarketcap. Rankings, Values and Statistics from Coinmarketcap.com. Available online: https://coinmarketcap.com/ (accessed on 21 December 2022).
2. Australian Competition and Consumer Commission. Targeting Scams. Report of the ACCC on Scams Activity 2018 (Issue May). 2019. Available online: https://www.mailguard.com.au/hubfs/ACCC%20on%20scams%20activity.pdf (accessed on 28 February 2024).
3. Krishnan, L.P.; Vakilinia, I.; Reddivari, S.; Ahuja, S. Scams and Solutions in Cryptocurrencies—A Survey Analyzing Existing Machine Learning Models. *Information* **2023**, *14*, 171. [CrossRef]
4. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *arXiv* **2017**, arXiv:1706.03762.
5. Kabla, A.H.H.; Anbar, M.; Manickam, S.; Karupayah, S. Eth-PSD: A machine learning-based phishing scam detection approach in ethereum. *IEEE Access* **2022**, *10*, 118043–118057. [CrossRef]
6. Wen, T.; Xiao, Y.; Wang, A.; Wang, H. A novel hybrid feature fusion model for detecting phishing scam on Ethereum using deep neural network. *Expert Syst. Appl.* **2023**, *211*, 118463. [CrossRef]
7. Gu, Z.; Lin, D.; Wu, J. On-chain analysis-based detection of abnormal transaction amount on cryptocurrency exchanges. *Phys. A Stat. Mech. Its Appl.* **2022**, *604*, 127799. [CrossRef]
8. Aziz, R.M.; Mahto, R.; Goel, K.; Das, A.; Kumar, P.; Saxena, A. Modified genetic algorithm with deep learning for fraud transactions of ethereum smart contract. *Appl. Sci.* **2023**, *13*, 697. [CrossRef]
9. Cui, W.; Gao, C. WTEYE: On-chain wash trade detection and quantification for ERC20 cryptocurrencies. *Blockchain Res. Appl.* **2023**, *4*, 100108. [CrossRef]
10. Wang, L.; Xu, M.; Cheng, H. Phishing scams detection via temporal graph attention network in Ethereum. *Inf. Process. Manag.* **2023**, *60*, 103412. [CrossRef]
11. Liu, Z.; Wang, Y.; Wang, S.; Zhao, X.; Wang, H.; Yin, H. Heterogeneous graphs neural networks based on neighbor relationship filtering. *Expert Syst. Appl.* **2024**, *239*, 122489. [CrossRef]
12. Shaw, P.; Uszkoreit, J.; Vaswani, A. Self-attention with relative position representations. *arXiv* **2018**, arXiv:1803.02155.
13. Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J.; Le, Q.V.; Salakhutdinov, R. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv* **2019**, arXiv:1901.02860.
14. Yang, D.; Lin, Y.; Wei, J.; Lin, X.; Zhao, X.; Yao, Y.; Tao, T.; Liang, B.; Lu, S.-G. Assisting Heart Valve Diseases Diagnosis via Transformer-Based Classification of Heart Sound Signals. *Electronics* **2023**, *12*, 2221. [CrossRef]
15. Chen, C.-F.R.; Fan, Q.; Panda, R. Crossvit: Cross-attention multi-scale vision transformer for image classification. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 357–366.
16. Wang, Y.; Chen, Y.; Wang, D. Convolution network enlightened transformer for regional crop disease classification. *Electronics* **2022**, *11*, 3174. [CrossRef]
17. Liu, M.; Ren, S.; Ma, S.; Jiao, J.; Chen, Y.; Wang, Z.; Song, W. Gated transformer networks for multivariate time series classification. *arXiv* **2021**, arXiv:2103.14438.
18. Huang, X.; Khetan, A.; Cvitkovic, M.; Karnin, Z. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv* **2020**, arXiv:2012.06678.
19. Li, L.; Jia, L.; Al Otaibi, S. Intelligent Recommendation Algorithm of Consumer Electronics Products with Graph Embedding and Multi-Head Self-Attention in IoE. *IEEE Trans. Consum. Electron.* **2023**, 10.1109/TCE.2023.3309978. [CrossRef]
20. Zhong, H.; Wang, M.; Zhang, X. HeMGNN: Heterogeneous Network Embedding Based on a Mixed Graph Neural Network. *Electronics* **2023**, *12*, 2124. [CrossRef]
21. Perozzi, B.; Al-Rfou, R.; Skiena, S. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 701–710.
22. Kurant, M.; Markopoulou, A.; Thiran, P. On the bias of BFS (breadth first search). In Proceedings of the 2010 22nd International Teletraffic Congress (lTC 22), Amsterdam, The Netherlands, 7–9 September 2010; pp. 1–8.
23. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.
24. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2980–2988.

25. Chen, L.; Peng, J.; Liu, Y.; Li, J.; Xie, F.; Zheng, Z. Phishing scams detection in ethereum transaction network. *ACM Trans. Internet Technol.* **2020**, *21*, 1–16. [CrossRef]
26. Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; Mei, Q. Line: Large-scale information network embedding. In Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 18–22 May 2015; pp. 1067–1077.
27. Grover, A.; Leskovec, J. 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, 13–17 August 2016; pp. 855–864.
28. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
29. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv* **2017**, arXiv:1710.10903.
30. Tan, R.; Tan, Q.; Zhang, Q.; Zhang, P.; Xie, Y.; Li, Z. Ethereum fraud behavior detection based on graph neural networks. *Computing* **2023**, *105*, 2143–2170. [CrossRef]
31. Du, X.; Yu, J.; Chu, Z.; Jin, L.; Chen, J. Graph autoencoder-based unsupervised outlier detection. *Inf. Sci.* **2022**, *608*, 532–550. [CrossRef]
32. Wang, D.; Cui, P.; Zhu, W. Structural deep network embedding. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 1225–1234.
33. Perozzi, B.; Kulkarni, V.; Chen, H.; Skiena, S. Don't walk, skip! online learning of multi-scale network embeddings. In Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017, Sydney, Australia, 31 July–3 August 2017; pp. 258–265.
34. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
35. Boser, B.E.; Guyon, I.M.; Vapnik, V.N. A training algorithm for optimal margin classifiers. In Proceedings of the Fifth Annual Workshop on Computational Learning Theory, Pittsburgh, PA, USA, 27–29 July 1992; pp. 144–152.
36. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
37. Liu, J.; Yin, C.; Wang, H.; Wu, X.; Lan, D.; Zhou, L.; Ge, C. Graph embedding-based money laundering detection for Ethereum. *Electronics* **2023**, *12*, 3180. [CrossRef]