IGT4ETH: An Isotropic Pre-trained Graph Transformer for Ethereum Account Classification

Ao Liu[1], Yanmei Zhang[1*], Youwei Wang[1], Qiang Duan[2]

[1]School of Information, Central University of Finance and Economics, Beijing, China
[2]Information Sciences & Technology Department, Pennsylvania State University, Abington PA, United States
2024110209@email.cufe.edu.cn, {Zhangym, wangyouwei}@cufe.edu.cn, qduan@psu.edu

Abstract

Pre-trained language models (PLMs) have shown strong potential in Ethereum account modeling and fraud detection. However, existing approaches often overlook the graph-structured nature of transaction networks. In addition, they struggle with the long-tail distribution of account activity, resulting in anisotropic embedding spaces and poor representation quality for low-frequency accounts. In this paper, we present IGT4ETH, a pre-trained Graph Transformer with an isotropy-enhanced processing, which explicitly models transaction topologies and mitigates representational anisotropy for Ethereum account classification. IGT4ETH improves structural representation by incorporating structural centrality and role embeddings into an Edge-augmented Graph Transformer, effectively capturing both topological and interaction patterns in transaction graphs. To further mitigate embedding anisotropy, we systematically evaluate various post-processing techniques. Among them, we adopt the Conceptor Negation (CN) method to softly suppress latent features dominated by high-frequency words via matrix conceptors, alongside a modified Focal-InfoNCE loss to enhance directional uniformity and representation balance. Extensive experiments on four real-world Ethereum account classification tasks, including phishing, exchange, mining, and ICO-wallet classification, demonstrate that IGT4ETH consistently outperforms state-of-the-art PLM-based baselines in terms of classification performance.

Code — https://github.com/Camus-Code/IGT4ETH

Introduction

Ethereum, the world's second-largest blockchain platform, has rapidly expanded thanks to its smart contract technology, attracting many users and supporting a large volume of on-chain assets. However, because accounts are pseudonymous and not linked to real-world identities, Ethereum offers user privacy but also opens the door to illegal activities such as phishing, money laundering, and Ponzi schemes (Hassan, Rehmani, and Chen 2022). As transaction activity on Ethereum becomes increasingly large-scale and complex, identifying suspicious behavior and understanding account patterns has become a critical challenge for blockchain security.

Recently, PLMs have shown strong performance in modeling Ethereum address behavior. Taking inspiration from natural language processing (NLP), transaction sequences are treated as sentences and addresses as tokens, allowing Transformer-based models to capture interaction patterns. Models like BERT4ETH (Hu et al. 2023) and ZipZap (Hu et al. 2024) have achieved promising results in tasks such as phishing detection and deanonymization.

Nevertheless, applying PLMs to large-scale on-chain transaction data presents two key challenges:

- **Insufficient utilization of structural information**: The Ethereum transaction network naturally forms a large and dynamic graph, where account interactions, graph structures, and edge features are essential for identifying account categories. However, existing models treat transactions merely as sequences, failing to fully utilize structural and edge-level information. This limits their ability to capture complex behaviors and detect more sophisticated fraud.
- **Anisotropy in embedding space**: Ethereum addresses follow a long-tail distribution, where a small number of high-frequency addresses dominate transactions, while the majority of low-frequency addresses lack sufficient samples, leading to poor embedding quality after model training. Prior studies have shown that embeddings often collapse into a narrow cone-like space (Gao et al. 2019), a phenomenon known as anisotropy (Li et al. 2020). This issue is prevalent in major pretrained models like BERT and GPT-2 (Ethayarajh 2019), and it reduces the expressiveness and generalization of address embeddings, especially for low-frequency accounts.

To address these challenges, we propose IGT4ETH, an isotropic pre-trained Graph Transformer for Ethereum account classification, which enhances structural awareness and embedding isotropy through two key strategies:

First, to better utilize structural information in Ethereum transaction graphs, IGT4ETH incorporates structural centrality and role embeddings to represent the importance and functional roles of addresses within the graph. Additionally, edge features are integrated via dedicated edge channels to guide the attention mechanism, enabling the model to focus

*Corresponding author.

on critical account interactions and capture complex behavioral patterns more effectively.

Second, to mitigate embedding anisotropy and representation degradation induced by the long-tail distribution of address frequencies, IGT4ETH applies post-processing techniques to promote isotropy in the representation space. This enhances the uniformity and generalizability of embeddings. Furthermore, a Focal-InfoNCE contrastive loss is incorporated during pretraining. Inspired by Focal Loss, it adaptively focuses more on hard negative samples that are similar to positives, while down-weighting easy negatives that are already distinguishable. This encourages the model to learn more refined and discriminative embeddings.

We evaluate IGT4ETH on multiple real-world Ethereum account classification tasks, including categories such as Phishing, Exchange, Mining, and ICO Wallet. Experimental results show that IGT4ETH consistently delivers high-quality embeddings and strong classification performance across diverse account types, demonstrating stable and reliable effectiveness.

The main contributions of this work are summarized as follows:

- To model transaction graph topology and account interactions, we augment the language model backbone with structural centrality, role embeddings, and an Edge-augmented Graph Transformer, enabling a more fine-grained understanding of account behaviors in the transaction graph.
- To mitigate embedding anisotropy, we apply isotropy-enhancing post-processing and incorporate a Focal-InfoNCE loss that emphasizes hard negatives, resulting in more isotropic and uniformly distributed embeddings.
- Experiments on four diverse Ethereum classification tasks show that IGT4ETH consistently improves embedding quality and classification performance across different settings.

Related work

Ethereum Representation Learning

Current approaches to Ethereum representation learning can be mainly divided into two categories: graph-based methods and language model-based methods.

Graph-based methods represent Ethereum transactions as graphs and extract structural features using graph walk algorithms or graph neural networks (GNNs). For instance, Trans2Vec (Wu et al. 2019) captures phishing behavior by incorporating transaction amounts and timestamps into biased random walks. (Beres et al. 2021) applied Diff2Vec and Role2Vec to identify account pairs controlled by the same entity. Other works formulate account classification as a subgraph-level task. (Liu et al. 2022) enhanced GNN expressiveness via neighbor filtering and feature engineering, and (Huang, Lin, and Wu 2022) combined multi-order GCN outputs to capture both local and global transaction dependencies.

Language model–based methods are inspired by the success of PLMs such as BERT in natural language processing. To adapt masked language modeling to the Ethereum domain, BERT4ETH (Hu et al. 2023) formulates a masked address prediction task using address sequences derived from transaction subgraphs. It achieves strong performance on phishing account detection, confirming the effectiveness of PLMs for blockchain modeling. Building on this, ZipZap (Hu et al. 2024) introduces a lightweight PLM for Ethereum, leveraging transaction dropping, Transformer layer sharing, and embedding compression to reduce model size and improve efficiency.

Post-processing Methods for Word Embeddings

To improve the expressiveness of address embeddings and reduce anisotropy in the embedding space, various post-processing methods have been proposed for static embeddings. (Levy, Goldberg, and Dagan 2015) introduced length normalization. (Sahlgren et al. 2016) applied mean centering. (Mu and Viswanath 2018) proposed removing the top principal components to reduce semantic-irrelevant noise in embeddings. Building on this work, (Liu, Ungar, and Sedoc 2019) further developed a post-processing method based on matrix conceptors, enabling the unsupervised suppression of high-variance latent features. (Liang et al. 2021) eliminated dominant directional components using weights learned from word similarity tasks.

For contextualized embeddings, (Ethayarajh 2019) revealed that models like BERT and GPT-2 exhibit pronounced anisotropy, with embeddings concentrated in narrow conical regions. To address this, (Rajaee and Pilehvar 2021) proposed a local cluster-based approach. (Sajjad et al. 2022) systematically compared four post-processing techniques and showed that these methods significantly enhance performance on lexical and sequence classification tasks, particularly when applied to higher layers of pretrained models, where richer semantic information is captured.

Graph Transformer

Graph Transformers combine the self-attention mechanism of Transformers with the structural modeling capabilities of graph learning, yielding strong performance and broad applicability across diverse graph-based tasks. They can generally be categorized into three types (Min et al. 2022): 1) GNNs as Auxiliary Modules, which embed GNN modules into the Transformer framework, effectively integrating the local structural modeling power of GNNs with the global modeling capability of Transformers. Representative models include UniMP (Shi et al. 2020), SGFormer (Wu et al. 2023b) and GraphGPS (Rampášek et al. 2022); 2) Improved Positional Embedding from Graphs, which generate enhanced positional embeddings based on graph structural features (e.g., degree, centrality) and add them to input representations before feeding them into a standard Transformer. Representative models include GraphiT (Mialon et al. 2021) and Graph-Bert (Zhang et al. 2020); 3) Improved Attention Matrices from Graphs, which introduce graph priors such as adjacency relations or shortest path distances as attention bias or masking mechanisms to strengthen structural awareness. Representative models include Graphormer (Ying et al. 2021) and EGT (Hussain, Zaki, and Subramanian 2021).

Methodology

Our proposed framework consists of five key stages: (a) Sequence and Graph Construction, (b) Node and Edge Embeddings, (c) Edge-augmented Graph Transformer Encoder, (d) Pretraining, and (e) Isotropy Enhancement, as illustrated in Figure 1(a)-(e).

Sequence and Graph Construction

For each central address, we construct its ego-subgraph—a directed graph that includes the central address and all its related transactional addresses. Each edge represents a transaction from sender to receiver, annotated with attributes like the total amount transferred and the number of interactions. This structure reflects both transaction frequency and fund flow patterns, providing a solid foundation for edge-level feature modeling.

We then generate a transaction sequence from the corresponding ego-subgraph, ordered in reverse chronological order to capture recent behavioral patterns. Each transaction includes features such as counterparty address, in-degree/out-degree, transaction amount, and transaction count.

To include the central address and reduce noise from repeated interactions, we follow the sequence construction of BERT4ETH (Hu et al. 2023). A dummy self-transaction with the target address and "Null" features is added at the sequence head as a global anchor for attention aggregation. Consecutive transactions with the same counterparty and direction within 72 hours are merged by keeping the earliest timestamp, summing amounts, and counting interactions.

Node and Edge Embeddings

To fully utilize transactional features in the Ethereum graph, we construct comprehensive embeddings at both node and edge levels. At the node level, embeddings capture static attributes, structural centrality, and structural roles, and at the edge level, embeddings characterize transactional intensity between account pairs, as detailed below.

Node Static Feature Embeddings To extract fundamental transactional properties of nodes and provide basic entity representations for the model, we define the static feature embedding as:

$$\boldsymbol{h}_i^{\text{feat}} = \boldsymbol{x}_i^{\text{addr}} + \boldsymbol{x}_i^{\text{amt}} + \boldsymbol{x}_i^{\text{cnt}} \tag{1}$$

where $\boldsymbol{h}_i^{\text{feat}} \in \mathbb{R}^d$ is the static feature embedding of node i with embedding dimension d. It is composed of three parts: the address embedding $\boldsymbol{x}_i^{\text{addr}} \in \mathbb{R}^d$, obtained by hashing the address to an integer index and retrieving its embedding vector from a lookup table; the transaction amount embedding $\boldsymbol{x}_i^{\text{amt}} \in \mathbb{R}^d$, generated by discretizing (binning) the transaction amounts involving the node within the sequence and mapping these bins to embeddings via a lookup table; and the transaction count embedding $\boldsymbol{x}_i^{\text{cnt}} \in \mathbb{R}^d$, which is formed by discretizing the counts of the node's transactions over a fixed time window and embedding these discrete values in the same manner.

Structural Centrality Embeddings In the Ethereum transaction network, different types of addresses (e.g., exchanges, ICO-wallets, phishing scam addresses) show significant variation in the ratio of outgoing to incoming transactions (Wu et al. 2023a). Such differences are reflected in the out-degree to in-degree ratio of each address within the transaction graph. To explicitly capture this topological information, we propose a new centrality embedding computed from the entire global transaction graph. For each address v_i ($v_i \in V$, where V is the set of all addresses), we calculate the ratio of its out-degree to total degree and map this value through a learnable embedding function $z(\cdot)$. The embedding is formally defined as:

$$\boldsymbol{h}_i^{\text{cent}} = z\left(\frac{\text{degree}_{\text{out}}(v_i)}{\text{degree}_{\text{in}}(v_i) + \text{degree}_{\text{out}}(v_i)}\right) \tag{2}$$

Here, $\text{degree}_{\text{in}}$ and $\text{degree}_{\text{out}}$ denote the node's in-degree and out-degree, respectively, and $z(\cdot)$ is a learnable embedding function that projects this structural ratio into a continuous vector space.

Structural Role Embeddings Different types of Ethereum accounts often play distinct structural roles in the transaction graph, reflected in their unique local connectivity patterns. To capture these structural features, we incorporate a structural role embedding module adapted from Graph-BERT (Zhang et al. 2020), which encodes each node's role based on its position-independent neighborhood structure.

Specifically, this module leverages the Weisfeiler-Lehman (WL) graph isomorphism test algorithm (Niepert, Ahmed, and Kutzkov 2016), which recursively hashes each node and its neighbors' labels to assign structural labels $WL(v_j)$ to nodes with similar local connectivity patterns. Following the positional encoding approach in Transformers (Vaswani et al. 2017), these discrete WL labels are then mapped into continuous structural embeddings:

$$\boldsymbol{h}_i^{\text{role}} = \left[\sin\left(\frac{WL(v_j)}{10000^{2l/d_h}}\right), \cos\left(\frac{WL(v_j)}{10000^{2l+1/d_h}}\right)\right]_{l=0}^{\lfloor d_h/2 \rfloor - 1} \tag{3}$$

where d_h is the embedding dimension and l indexes the embedding dimensions.

Finally, the initial representation of each address $\boldsymbol{h}_i^{(0)}$ is formed by summing the above three embeddings:

$$\boldsymbol{h}_i^0 = \boldsymbol{h}_i^{\text{feat}} + \boldsymbol{h}_i^{\text{role}} + \boldsymbol{h}_i^{\text{cent}} \tag{4}$$

Edge Embeddings To capture interaction features between account pairs in the transaction graph, we compute edge embeddings that represent transactional relationships between nodes. For each directed edge from node i to node j, the edge embedding is defined as:

$$\boldsymbol{e}_{ij}^0 = \boldsymbol{e}_{ij}^{\text{amt}} + \boldsymbol{e}_{ij}^{\text{cnt}} \tag{5}$$

Here, $\boldsymbol{e}_{ij}^0 \in \mathbb{R}^d$ is the initial embedding for the transaction edge from account i to account j. It combines two components: $\boldsymbol{e}_{ij}^{\text{amt}}$ and $\boldsymbol{e}_{ij}^{\text{cnt}}$, representing the embeddings of discretized transaction amounts and counts, respectively. Each

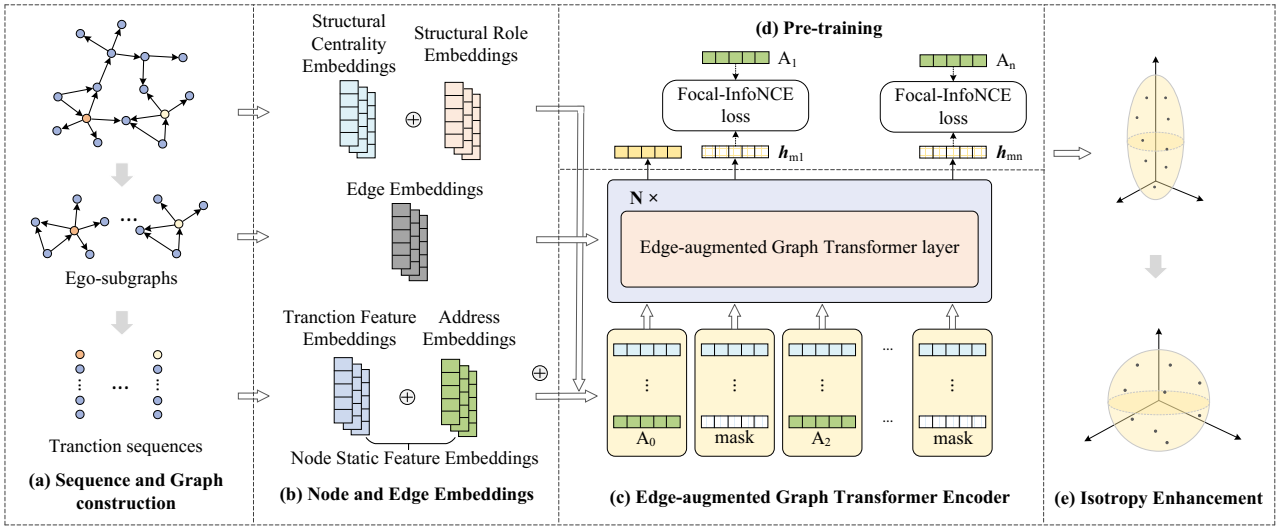


Figure 1: Overall architecture of the proposed model.

is mapped from its discrete bin to a trainable vector via a lookup table. Together, these embeddings capture both the volume and intensity of transactions on each edge.

Stacking all node embeddings produces the node input matrix for the transaction ego-subgraph, denoted as $\boldsymbol{H}^0 = [\boldsymbol{h}_1^0, \boldsymbol{h}_2^0, \ldots, \boldsymbol{h}_N^0]^\top \in \mathbb{R}^{N\times d}$, where N is the number of nodes and d is the embedding dimension. Similarly, aggregating all initial edge embeddings forms the edge feature matrix $\boldsymbol{E}^0 = [\boldsymbol{e}_{ij}^0]_{(i,j)\in\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}|\times d}$, where $\mathcal{E}$ denotes the set of edges. These matrices provide the node and edge features for the subsequent Graph Transformer model.

Edge-augmented Graph Transformer Encoder

The Ethereum transaction graph contains rich edge features, such as transaction amounts and frequencies, which are critical for effective account representation learning. Therefore, it is essential for the model to not only capture hierarchical node representations but also fully exploit edge information. To this end, we adopt a stacked Edge-augmented Graph Transformer (EGT) (Hussain, Zaki, and Subramanian 2021) encoder, which extends the standard Transformer by adding dedicated edge channels to directly model and propagate edge information in transaction graphs. The overall architecture is shown in Figure 1(c-sub).

The EGT encoder takes as input the initial node embedding matrix $\boldsymbol{H}^0 \in \mathbb{R}^{N\times d}$ and edge embedding matrix $\boldsymbol{E}^0 \in \mathbb{R}^{|\mathcal{E}|\times d}$, and updates them over L layers to produce the final node and edge representations $\boldsymbol{H}^L$ and $\boldsymbol{E}^L$.

For the l-th layer and the k-th attention head, the attention computation is defined as:

$$\mathrm{Attn}(\boldsymbol{Q}_{k,\ell}, \boldsymbol{K}_{k,\ell}, \boldsymbol{V}_{k,\ell}) = \boldsymbol{A}_{k,\ell}\boldsymbol{V}_{k,\ell} \tag{6}$$

$$\boldsymbol{A}_{k,\ell} = \mathrm{softmax}(\boldsymbol{H}_{k,\ell}) \odot \sigma(\boldsymbol{G}^e_{k,\ell}) \tag{7}$$

$$\boldsymbol{H}_{k,\ell} = \mathrm{clip}\left(\frac{\boldsymbol{Q}_{k,\ell}\boldsymbol{K}^\top_{k,\ell}}{\sqrt{d_k}}\right) + \boldsymbol{E}^e_{k,\ell} \tag{8}$$

Here, $\boldsymbol{Q}_{k,\ell}$, $\boldsymbol{K}_{k,\ell}$, and $\boldsymbol{V}_{k,\ell}$ denote the query, key, and value vectors of the k-th head in layer ℓ, respectively; $\boldsymbol{E}^e_{k,\ell}$ is a learnable edge bias term; $\boldsymbol{G}^e_{k,\ell}$ is an edge gating function; $\sigma(\cdot)$ is a sigmoid activation; and $\mathrm{clip}(\cdot)$ restricts values to the range $[-5, 5]$ to ensure numerical stability.

Within each layer, normalization is applied to the node and edge representations prior to the residual connections, following the Pre-Norm design.

$$\hat{\boldsymbol{h}}_i^\ell = \mathrm{LN}(\boldsymbol{h}_i^{\ell-1}), \quad \hat{\boldsymbol{e}}_{ij}^\ell = \mathrm{LN}(\boldsymbol{e}_{ij}^{\ell-1}) \tag{9}$$

The updated node and edge representations are then computed as follows:

$$\hat{\hat{\boldsymbol{h}}}_i^\ell = \boldsymbol{h}_i^{\ell-1} + \boldsymbol{O}_h^\ell \Big\|_{k=1}^{H} \sum_{j=1}^{N} \boldsymbol{A}_{k,\ell}^{ij} \boldsymbol{V}_{k,\ell} \hat{\boldsymbol{h}}_j^\ell \tag{10}$$

$$\hat{\hat{\boldsymbol{e}}}_{ij}^\ell = \boldsymbol{e}_{ij}^{\ell-1} + \boldsymbol{O}_e^\ell \Big\|_{k=1}^{H} \boldsymbol{H}_{k,\ell}^{ij} \tag{11}$$

where $\boldsymbol{O}_h^\ell$ and $\boldsymbol{O}_e^\ell$ are the output projection matrices for nodes and edges, respectively; H is the number of attention heads; and $\|$ denotes concatenation.

Subsequently, the node and edge representations are processed by their respective feed-forward networks:

$$\boldsymbol{h}_i^\ell = \hat{\hat{\boldsymbol{h}}}_i^\ell + \mathrm{FFN}_h^\ell\big(\mathrm{LN}(\hat{\hat{\boldsymbol{h}}}_i^\ell)\big), \quad \boldsymbol{e}_{ij}^\ell = \hat{\hat{\boldsymbol{e}}}_{ij}^\ell + \mathrm{FFN}_e^\ell\big(\mathrm{LN}(\hat{\hat{\boldsymbol{e}}}_{ij}^\ell)\big) \tag{12}$$

where $\mathrm{FFN}_h^\ell(\cdot)$ and $\mathrm{FFN}_e^\ell(\cdot)$ are standard two-layer feed-forward networks with non-linear activations, applied to node and edge representations, respectively.

Finally, we apply layer normalization to obtain the final node and edge embeddings:

$$\boldsymbol{h}_i^L = \mathrm{LN}(\boldsymbol{h}_i^L), \quad \boldsymbol{e}_{ij}^L = \mathrm{LN}(\boldsymbol{e}_{ij}^L) \tag{13}$$

Pre-training

We adopt contrastive learning in the pre-training phase to model Ethereum transaction behaviors. During training, certain addresses are randomly masked, and the model learns to recover their original embeddings based on surrounding context, effectively capturing both structural and contextual information. Given the vast number of Ethereum addresses, contrastive learning improves training efficiency by comparing positive and negative pairs, without relying on full-address classification as required in BERT (Devlin et al. 2019).

The common InfoNCE loss (Lin et al. 2017), used by Bert4ETH (Hu et al. 2023) and Zipzap (Hu et al. 2024), treats all samples equally, making it difficult to handle noisy positives and hard negatives that are semantically similar to the anchor but belong to different classes. To address this, we adopt a Focal-InfoNCE loss (Cheng et al. 2023) that adaptively adjusts sample weights: it down-weights low-quality positive samples that are inconsistent with the context, and up-weights hard negatives that have high similarity to the anchor, encouraging the model to focus on more informative contrastive signals. This loss not only enhances representation discrimination but also alleviates anisotropy by distinguishing hard negatives.

$$\mathcal{L} = -\frac{1}{|\mathbb{M}|} \sum_{i \in \mathbb{M}} \log \frac{\exp(s_i^+)}{\exp(s_i^+) + \sum_{j \in \mathbb{N}} \exp(s_{ij}^-)} \tag{14}$$

$$s_i^+ = \frac{\left(\boldsymbol{h}_i^\top \tilde{\boldsymbol{h}}_i\right)^2}{\tau}, \quad s_{ij}^- = \frac{\left(\boldsymbol{h}_i^\top \tilde{\boldsymbol{h}}_j\right)\left(\boldsymbol{h}_i^\top \tilde{\boldsymbol{h}}_j + \mu\right)}{\tau} \tag{15}$$

where $\mathbb{M}$ is the set of masked addresses in the sequence, $\boldsymbol{h}_i$ denotes the contextual embedding of masked address i, and $\tilde{\boldsymbol{h}}_i$ is its original (unmasked) embedding used as the positive sample. $\mathbb{N}$ represents the set of negative samples obtained via sampling, with $\tilde{\boldsymbol{h}}_j$ being their embeddings. τ is the temperature hyperparameter, and μ controls the weighting of hard negatives.

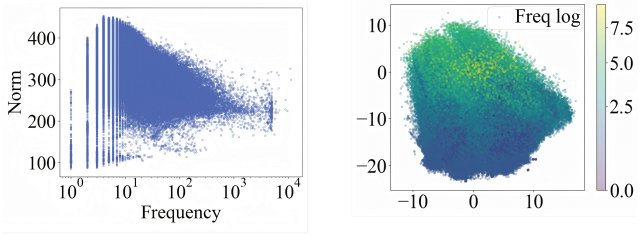
Isotropy Enhancement

The Ethereum transaction network exhibits a clear power-law (long-tail) distribution, where a small fraction of addresses dominate most transactions (Hu et al. 2023). This imbalance induces anisotropy in the embedding space. As shown in Figure 2a, embedding vector norms negatively correlate with address frequency: high-frequency addresses tend to have smaller norms and more centralized embeddings, whereas low-frequency addresses yield larger norms and more scattered embeddings.

To further analyze this embedding structure, we apply principal component analysis (PCA) to the address embeddings and project them onto the top two principal components (Figure 2b), where brighter colors denote higher frequency. The resulting conical distribution shows high-frequency addresses clustering near the origin, while low-frequency ones spread outward—reflecting the anisotropic geometry. Such a layout can cause semantically distinct addresses to exhibit overly similar embeddings in Euclidean space, harming embedding quality.

Metric	Phishing	ICO-Wallet	Mining	Exchange	Normal
# Ctr Addr	3,220	167	103	353	594,038
# All Addr	151,415	29,885	62,118	488,423	2,609,855
# Txns	328,261	68,430	249,144	1,175,319	11,350,640
# In Txns	182,356	54,932	3,045	375,406	3,159,707
# Out Txns	145,905	13,498	246,099	799,913	8,190,933

Table 1: Dataset Statistics



(a) Norm–frequency correlation (b) PCA projection

Figure 2: (a) the correlation between embedding vector norms and address frequency, (b) the projection of embedding distribution onto the top two principal components, where brightness indicates address frequency.

To mitigate anisotropy, we apply post-processing methods to enhance directional uniformity and representation balance. Drawing from methods used in natural language processing for static and contextual embeddings, We systematically evaluate six post-processing methods on the outputs of different layers of the IGT4ETH model, including z-score standardization, min-max scaling, unit length normalization (ulen) (Levy, Goldberg, and Dagan 2015), All-but-the-Top (abtt) (Mu and Viswanath 2018), Conceptor Negation (CN) (Liu, Ungar, and Sedoc 2019), and a cluster-based method (Rajaee and Pilehvar 2021). Our experiments show that all methods alleviate embedding anisotropy to varying extents, with CN providing the greatest gains in uniformity and discriminative power.

Experimental Setup

Datasets Building upon the Bert4ETH dataset, which contains 3,220 phishing and 594,038 normal accounts, we extend it by collecting more labeled samples from Xblock and Etherscan, following the data acquisition method described in (Jin et al. 2025). The extended dataset adds 173 ICO-wallet, 127 mining, and 374 exchange accounts. To ensure temporal consistency, all transactions involving these accounts from January 1, 2017 to May 1, 2022 were gathered. Detailed statistics are shown in Table 1.

Baselines Our experiments compare four categories of baseline methods: 1) graph representation learning algorithms, including DeepWalk (Perozzi, Al-Rfou, and Skiena

2014), Trans2Vec (Wu et al. 2019), Diff2Vec (Rozemberczki and Sarkar 2018), and Role2Vec (Ahmed et al. 2022), where Trans2Vec is specifically tailored for phishing account detection; 2) graph neural networks (GNNs), including GCN (Kipf and Welling 2017), GraphSAGE (Hamilton, Ying, and Leskovec 2017), GAT (Veličković et al. 2018), and GIN (Xu et al. 2019); 3) pretrained language models, including BERT4ETH (Hu et al. 2023) and ZipZap (Hu et al. 2024), IGT4ETH and its isotropy-enhanced variant IGT4ETH*; 4) Graph Transformer models, including UniMP (Shi et al. 2020), Graphormer (Ying et al. 2021), GraphGPS (Rampášek et al. 2022), SGFormer (Wu et al. 2023b), Exphormer (Shirzad et al. 2023), and Polynormer (Deng, Yue, and Zhang 2024), which are compared in the section on Graph Transformer methods.

Implementation Details For graph learning methods, we follow the self-supervised training scheme introduced in DeepWalk, setting the number of random walks per node to 10, walk length to 20, and context window size to 5. For GNN models, we employ two graph layers with a neighbor sampling size fixed at 50. For BERT4ETH and ZipZap, we follow the original configurations, setting the maximum transaction sequence lengths to 100 and 29, respectively. For Graph Transformer models, SGFormer uses 8 Transformer layers and 2 GNN layers, while all other models are configured with 8 graph Transformer layers. For IGT4ETH, we set the number of EGT layers to 8, with 8 attention heads per layer and a maximum input sequence length of 100. Across all models, the masking ratio is set to 0.8 during pre-training to avoid label leakage, while a two-layer fully connected layer is employed in the fine-tuning and fixed-training phases. Additionally, The negative-to-positive sample ratio is fixed at 100:1 across the four binary classification tasks, with the hidden dimension size set to 64 and the batch size set to 256. We report the average F_1 score over five runs as the final evaluation metric, where for each model and account category, the threshold is selected to maximize the F_1 score. Experiments are conducted on a system equipped with two NVIDIA RTX 3090 GPUs, each with 24GB memory.

Evaluation on Account Classification

We systematically evaluate IGT4ETH on four Ethereum account classification tasks under two training strategies: fixed training and fine-tuning.

Under the fixed-training setting (Table 2), GNN-based methods generally outperform random-walk-based ones, with GAT achieving the best GNN performance, highlighting the value of explicit graph structure modeling. Pretrained language models further improve performance, with IGT4ETH consistently achieving the best results across all tasks. Compared to the second-best ZipZap, IGT4ETH boosts F_1 scores by 4.82%, 7.48%, 6.50%, and 6.03% on Phishing, Exchange, Mining, and ICO-Wallet, respectively. These improvements highlight that the full utilization of graph structure and transactional edge information can significantly enhance the expressive capacity of address representations. The enhanced IGT4ETH* further improves performance, confirming the effectiveness of the post-processing method.

Methods	Phishing	Exchange	Mining	ICO-Wallet
DeepWalk	26.36	55.12	45.62	24.98
Trans2Vec	24.42	42.14	36.36	22.02
Diff2Vec	27.17	48.14	33.33	22.65
Role2Vec	30.74	38.05	37.04	16.90
GraphSAGE	31.17	66.33	39.22	56.84
GCN	36.68	52.31	62.30	60.47
GAT	37.50	63.49	67.80	66.67
GIN	32.24	55.96	51.72	71.11
BERT4ETH	55.02	69.89	71.43	70.65
ZipZap	53.94	79.65	75.71	76.95
IGT4ETH	58.76	87.13	82.21	82.98
IGT4ETH*	**61.14**	**88.21**	**84.21**	**86.32**

Table 2: F_1 score (%) comparison of different methods on various account types under fixed-training.

Methods	Phishing	Exchange	Mining	ICO-Wallet
Fine-tuning (with pre-training)				
BERT4ETH	62.37	84.91	73.68	72.92
ZipZap	63.81	85.02	89.10	80.00
IGT4ETH	**65.99**	**87.92**	**92.06**	**88.66**
Without pre-training				
BERT4ETH	47.44	40.29	62.96	56.84
ZipZap	48.65	69.77	69.21	68.75
IGT4ETH	**51.84**	**77.64**	**76.92**	**76.06**

Table 3: F_1 score (%) comparison under fine-tuning and without pre-training.

Under fine-tuning (Table 3), IGT4ETH surpasses BERT4ETH and ZipZap across all tasks and outperforms its fixed-training variant, validating the benefits of fine-tuning. Removing pretraining leads to notable performance degradation, emphasizing its essential role in model generalization.

Ablation Study

We conduct ablation studies under fixed-training settings to assess the contribution of five core components in IGT4ETH, which refers to the base model without the isotropy enhancement module (w/o CN). Each component is removed individually to evaluate performance (Table 4).

Removing transaction edge embeddings and the dedicated edge channel (w/o edge) leads to notable F_1 drops of 4.05%, 5.27%, 8.24%, and 6.79% on Phishing, Exchange, Mining, and ICO-Wallet tasks, respectively, highlighting the importance of interaction-level information. Excluding structural centrality embeddings (w/o cent) causes large declines, especially on Exchange and ICO-Wallet tasks (up to 7.38% and 4.96%), showing the value of global node importance. Structural role embeddings (w/o role) also bring consistent but smaller drops, confirming the usefulness of role-aware features. Replacing Focal-InfoNCE with standard InfoNCE (w/o Focal) results in performance degradation, particularly on Phishing and ICO-Wallet, validating the benefit of em-

Methods	Phishing	Exchange	Mining	ICO-Wallet
w/o edge	54.71	81.86	73.97	75.02
w/o cent	55.64	80.75	79.43	78.02
w/o role	56.32	85.05	79.25	81.65
w/o Focal	55.22	83.67	81.63	79.06
w/o CN (IGT4ETH)	58.76	87.13	82.21	82.98
IGT4ETH*	**61.14**	**88.21**	**84.21**	**86.32**

Table 4: Ablation study results in terms of F_1 score (%).

Methods	Phishing	Exchange	Mining	ICO-Wallet
UniMP	43.28	64.58	66.67	46.81
SGFormer	51.84	58.97	71.30	66.67
Exphormer	53.85	64.29	70.65	73.68
Graphormer	55.02	75.80	79.31	78.85
Polynormer	56.14	78.27	74.19	77.51
GraphGPS	56.66	80.42	67.86	80.06
IGT4ETH	**58.76**	**87.13**	**82.21**	**82.98**

Table 5: Performance comparison of different Graph Transformer models in terms of F_1 score (%).

phasizing hard negatives. Finally, adding the isotropy enhancement module (CN) improves F_1 scores across all tasks, verifying that post-processing improves the quality of address embeddings and overall generalization.

Comparison with Different Graph Transformer Methods

Table 5 compares IGT4ETH with representive Graph Transformer models, including UniMP, SGFormer, Graphormer, GraphGPS, SGFormer, and Exphormer, under the fixed-training setting. IGT4ETH achieves the highest F_1 scores across all four tasks, notably outperforming others by 6.71% and 2.90% on Exchange and Mining, respectively. This highlights its strength in modeling transaction behavior and structural features.

Although Polynormer and GraphGPS perform well on some tasks, they struggle with the complexity of Ethereum's evolving transaction graph. By jointly leveraging centrality, role, and edge information, IGT4ETH more effectively captures topological roles and transactional interactions, giving it a clear advantage in blockchain account classification.

Analysis of Isotropy Enhancement Methods

To evaluate the impact of isotropy enhancement, we compare various embedding post-processing methods on Ethereum account classification (Table 6). CN method (IGT4ETH*) achieves the highest F_1 scores on most tasks, including 88.21% on Exchange and 86.32% on ICO-Wallet, outperforming abtt, z-score, and cluster-based methods. This demonstrates CN's effectiveness in improving embedding quality. Among other methods, abtt achieves the best score on Mining (83.33%), while z-score and cluster-based methods perform consistently well on Exchange and ICO-Wallet, indicating the general benefit of isotropy enhancement when applied properly.

Methods	Phishing	Exchange	Mining	ICO-Wallet
IGT4ETH	58.76	87.13	82.21	82.98
abtt	60.35	86.87	**83.33**	84.21
zscore	60.62	87.88	82.54	85.42
ulen	57.09	81.87	82.76	71.43
min-max	51.91	70.18	0	0
cluster-based	59.95	87.80	78.69	84.21
CN (IGT4ETH*)	**61.14**	**88.21**	82.76	**86.32**

Table 6: Performance comparison of different isotropy enhancement methods in terms of F_1 score (%).

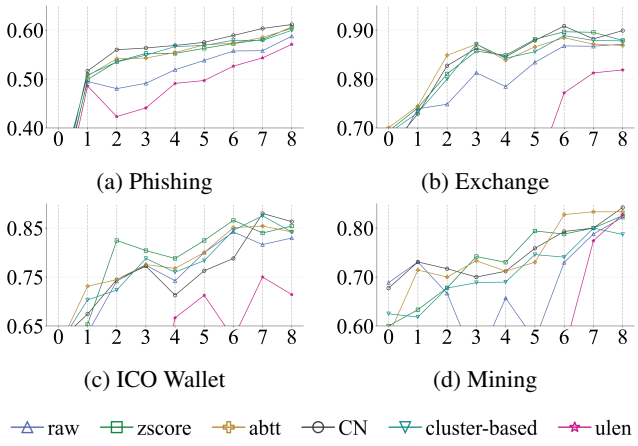


Figure 3: Effect of post-processing on layer-wise representations across four tasks. The x-axis denotes the model layer index, and the y-axis represents the F_1 score (%).

Figure 3 illustrate how various isotropy enhancement methods affect layer-wise F_1 scores across four tasks, with the min-max method excluded due to consistently poor performance. As shown, the CN method maintains stable, high performance across layers, while the unprocessed model fluctuates. Though some methods perform well on certain tasks, they exhibit instability across categories. Overall, the CN method not only mitigates embedding anisotropy but also leads to more stable intermediate representations.

Conclusion

We present IGT4ETH, a pre-trained graph Transformer that learns structurally-aware and isotropic address representations for Ethereum account classification. It addresses limitations in structural modeling and embedding anisotropy found in existing language models. By incorporating structural centrality, role embeddings, and Graph Transformer, the model better captures transaction structures and account interactions. Additionally, isotropy-enhancing post-processing and a Focal-InfoNCE loss mitigate embedding degradation from long-tail distributions, improving the account representation quality. Experiments on multiple classification tasks confirm the effectiveness of these strategies, highlighting IGT4ETH's strong performance in Ethereum account classification.

Acknowledgments

This work was jointly supported by the National Key Research and Development Program of China under Grant 2024YFC3308100, the National Natural Science Foundation of China under Grant 62372493, and the High-Performance Computing Platform of Central University of Finance and Economics.

References

Ahmed, N. K.; Rossi, R. A.; Lee, J. B.; Willke, T. L.; Zhou, R.; Kong, X.; and Eldardiry, H. 2022. Role-Based Graph Embeddings. *IEEE Transactions on Knowledge and Data Engineering*, 34(5): 2401–2415.

Beres, F.; Seres, I. A.; Benczur, A. A.; and Quintyne-Collins, M. 2021. Blockchain is Watching You: Profiling and Deanonymizing Ethereum Users . In *2021 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, 69–78.

Cheng, Q.; Yang, X.; Sun, T.; Li, L.; and Qiu, X. 2023. Improving Contrastive Learning of Sentence Embeddings from AI Feedback. In *Findings of the Association for Computational Linguistics: ACL 2023*, 11122–11138.

Deng, C.; Yue, Z.; and Zhang, Z. 2024. Polynormer: Polynomial-Expressive Graph Transformer in Linear Time. In *The Twelfth International Conference on Learning Representations*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 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, volume 1 (long and short papers)*, 4171–4186.

Ethayarajh, K. 2019. How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 55–65.

Gao, J.; He, D.; Tan, X.; Qin, T.; Wang, L.; and Liu, T. 2019. Representation Degeneration Problem in Training Natural Language Generation Models. In *International Conference on Learning Representations*.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 1025–1035.

Hassan, M. U.; Rehmani, M. H.; and Chen, J. 2022. Anomaly detection in blockchain networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 25(1): 289–318.

Hu, S.; Huang, T.; Chow, K.-H.; Wei, W.; Wu, Y.; and Liu, L. 2024. Zipzap: Efficient training of language models for large-scale fraud detection on blockchain. In *Proceedings of the ACM Web Conference 2024*, 2807–2816.

Hu, S.; Zhang, Z.; Luo, B.; Lu, S.; He, B.; and Liu, L. 2023. Bert4eth: A pre-trained transformer for ethereum fraud detection. In *Proceedings of the ACM Web Conference 2023*, 2189–2197.

Huang, T.; Lin, D.; and Wu, J. 2022. Ethereum Account Classification Based on Graph Convolutional Network. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 69(5): 2528–2532.

Hussain, M. S.; Zaki, M. J.; and Subramanian, D. 2021. Global Self-Attention as a Replacement for Graph Convolution. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Jin, C.; Zhou, J.; Xie, C.; Yu, S.; Xuan, Q.; and Yang, X. 2025. Enhancing Ethereum Fraud Detection via Generative and Contrastive Self-Supervision. *IEEE Transactions on Information Forensics and Security*, 20: 839–853.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

Levy, O.; Goldberg, Y.; and Dagan, I. 2015. Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association for Computational Linguistics*, 3: 211–225.

Li, B.; Zhou, H.; He, J.; Wang, M.; Yang, Y.; and Li, L. 2020. On the Sentence Embeddings from Pre-trained Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 9119–9130.

Liang, Y.; Cao, R.; Zheng, J.; Ren, J.; and Gao, L. 2021. Learning to remove: Towards isotropic pre-trained bert embedding. In *International conference on artificial neural networks*, 448–459.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017. Focal Loss for Dense Object Detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2999–3007.

Liu, J.; Zheng, J.; Wu, J.; and Zheng, Z. 2022. FA-GNN: Filter and Augment Graph Neural Networks for Account Classification in Ethereum. *IEEE Transactions on Network Science and Engineering*, 9(4): 2579–2588.

Liu, T.; Ungar, L.; and Sedoc, J. 2019. Unsupervised Post-Processing of Word Vectors via Conceptor Negation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01): 6778–6785.

Mialon, G.; Chen, D.; Selosse, M.; and Mairal, J. 2021. Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667*.

Min, E.; Chen, R.; Bian, Y.; Xu, T.; Zhao, K.; Huang, W.; Zhao, P.; Huang, J.; Ananiadou, S.; and Rong, Y. 2022. Transformer for graphs: An overview from architecture perspective. *arXiv preprint arXiv:2202.08455*.

Mu, J.; and Viswanath, P. 2018. All-but-the-Top: Simple and Effective Postprocessing for Word Representations. In *International Conference on Learning Representations*.

Niepert, M.; Ahmed, M.; and Kutzkov, K. 2016. Learning Convolutional Neural Networks for Graphs. In *Proceedings*

of The 33rd International Conference on Machine Learning, volume 48 of *Proceedings of Machine Learning Research*, 2014–2023.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. DeepWalk: online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 701–710.

Rajaee, S.; and Pilehvar, M. T. 2021. A Cluster-based Approach for Improving Isotropy in Contextual Embedding Space. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 575–584.

Rampášek, L.; Galkin, M.; Dwivedi, V. P.; Luu, A. T.; Wolf, G.; and Beaini, D. 2022. Recipe for a general, powerful, scalable graph transformer. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*.

Rozemberczki, B.; and Sarkar, R. 2018. Fast sequence-based embedding with diffusion graphs. In *International workshop on complex networks*, 99–107.

Sahlgren, M.; Gyllensten, A. C.; Espinoza, F.; Hamfors, O.; Karlgren, J.; Olsson, F.; Persson, P.; Viswanathan, A.; and Holst, A. 2016. The Gavagai Living Lexicon. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*, 344–350.

Sajjad, H.; Alam, F.; Dalvi, F.; and Durrani, N. 2022. Effect of post-processing on contextualized word representations. In *Proceedings of the 29th International Conference on Computational Linguistics*, 3127–3142.

Shi, Y.; Huang, Z.; Feng, S.; Zhong, H.; Wang, W.; and Sun, Y. 2020. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*.

Shirzad, H.; Velingker, A.; Venkatachalam, B.; Sutherland, D. J.; and Sinop, A. K. 2023. EXPHORMER: sparse transformers for graphs. In *Proceedings of the 40th International Conference on Machine Learning*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Wu, J.; Huang, B.; Liu, J.; Li, Q.; and Zheng, Z. 2023a. Understanding the dynamic and microscopic traits of typical Ethereum accounts. *Information Processing & Management*, 60(4): 103384.

Wu, J.; Yuan, Q.; yan Lin, D.; You, W.; Chen, W.; Chen, C.; and Zheng, Z. 2019. Who Are the Phishers? Phishing Scam Detection on Ethereum via Network Embedding. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52: 1156–1166.

Wu, Q.; Zhao, W.; Yang, C.; Zhang, H.; Nie, F.; Jiang, H.; Bian, Y.; and Yan, J. 2023b. SGFormer: simplifying and empowering transformers for large-graph representations. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*.

Ying, C.; Cai, T.; Luo, S.; Zheng, S.; Ke, G.; He, D.; Shen, Y.; and Liu, T.-Y. 2021. Do transformers really perform bad for graph representation? In *Proceedings of the 35th International Conference on Neural Information Processing Systems*.

Zhang, J.; Zhang, H.; Xia, C.; and Sun, L. 2020. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*.