A Background

In this section, we provide an overview of blockchain technology and cryptocurrency, laying the groundwork for understanding the subsequent discussions in this paper.

Blockchain and Cryptocurrency. Blockchain technology has gained growing attention recently for its strong security features and decentralized structure. It is characterized by a sequence of cryptographically secured blocks that operate on a network of nodes [42]. This design ensures data immutability and verifiability while allowing universal access, enabling participants to interact with the ledger from anywhere at any time. Once recorded on the ledger, transactions become irreversible and are executed securely and transparently, which helps safeguard the integrity of data exchanges.

With the support of blockchain technology, cryptocurrencies have surged in popularity as an innovative means of conducting secure digital transactions. Unlike traditional currencies, cryptocurrencies operate without a centralized authority and are managed through decentralized systems. This decentralization maintains participant anonymity, offering robust privacy protection; however, it complicates efforts to identify fraudulent activities within the market.

Blockchain Models and EVM Chains. Various operational models exist within blockchain technology. For instance, Bitcoin, the first cryptocurrency network, operates using the Unspent Transaction Output (UTXO) model [66]. In this model, each transaction utilizes unspent outputs from previous transactions as inputs, generating new unspent outputs for subsequent transactions. This method preserves transaction integrity by streamlining ownership verification and enhancing security measures related to transaction immutability.

In contrast, the Ethereum Virtual Machine (EVM) introduced an account-based model, akin to traditional banking systems, where balances are maintained in user accounts [67]. This model enables direct value transfer and supports advanced features such as smart contracts, which are self-executing agreements with terms embedded directly within the blockchain. Due to its versatility and strong developer support, the EVM has become the standard for building blockchain networks and decentralized applications. The three notable EVM-based networks discussed in this work are Ethereum, Polygon, and Binance Smart Chain [68]:

- *Ethereum*, the pioneering EVM chain, has developed a robust platform for decentralized applications. It supports a wide range of decentralized services, from financial transactions to games and autonomous organizations. Its native token, Ether, holds the second-largest market capitalization, second only to Bitcoin.
- *Polygon* enhances Ethereum's functionality as an EVM-compatible chain by offering faster transactions and reduced fees. Functioning as a sidechain to Ethereum, it addresses scalability issues with a multi-chain infrastructure, which is particularly advantageous for developers seeking efficient transaction throughput within the Ethereum ecosystem.
- *Binance Smart Chain* provides a similar EVM-compatible environment with a focus on scalability and user experience. It has carved out a niche by emphasizing rapid transactions and minimal fees, particularly attracting decentralized finance (DeFi) applications and NFTs.

ERC20 and BEP20 Standards. The ERC20 standard defines a framework for fungible tokens on the Ethereum blockchain. These fungible tokens are digital assets that are identical in type and value, making them interchangeable with one another. This standardization simplifies the process of trading and exchanging tokens and enhances their interoperability across various applications. Similarly, BEP20 is a standard used on the Binance Smart Chain (BSC), mirroring many of the functionalities of ERC20 while optimizing for faster transactions and lower fees.

Accounts and Transactions. EVM-compatible chains typically support two principal types of accounts: External Owned Accounts (EOAs) and smart contracts. EOAs function much like traditional bank accounts, as they are directly managed by users through a private key, granting them full autonomy over transactions. In contrast, smart contracts are autonomous programs that reside on the blockchain and execute automatically when predefined conditions are met. These programs are crucial for a variety of operations on EVM chains, from facilitating transactions in the token markets to managing decentralized finance (DeFi) protocols and automated governance mechanisms.

A transaction includes various details, such as the sender's and recipient's actions, signature, nonce, data, gas limit, maximum priority fee per gas, and maximum fee per gas. In the token market, these transactions facilitate diverse blockchain events like token issuance and transfers. This architectural framework not only supports complex financial interactions but also enhances security across the blockchain ecosystem.

B Supplemental Related Work

Graphs-of-Graphs Analysis. The analysis of Graphs-of-Graphs (GoG) systems has become a crucial method for understanding complex relationships within and across different network layers in various domains. For instance, Chen et al. [69] examined the dynamics of event propagation on social platforms like Twitter. They analyzed follower link roles by grouping users based on their language settings, treating these groups as local graphs, with following or retweeting relationships represented as edges. Similarly, Wang et al. [70] modeled intra-level and inter-level causal relationships within interdependent networks, effectively tracing and identifying root causes in complex interconnected system structures. In more specialized applications, Liu et al. [71] employed GoG to enhance hazard identification at construction sites. They mapped interactions between characters and hazard networks, simplifying complex network structures to improve safety outcomes. Additionally, Chen et al. [72, 73] investigated the manipulation of connectivity in multi-layered networks, uncovering the structural dynamics that govern these complex systems. These studies underscore the powerful capability of GoG analysis in providing a deeper understanding of intricate graph systems.

C Basic Structure Properties

In this section, we explore several fundamental graph properties relevant to our analysis, as discussed in subsection 4.1 and subsection 5.2. We measure seven key graph properties: the number of nodes, the number of edges, density, assortativity, reciprocity, clustering coefficient, and effective diameter. These properties provide a comprehensive structural overview of the graph, which is essential for understanding its characteristics and implications in the context of token transfer networks.

First, we consider the number of nodes and edges, which quantitatively describe the scale and potential complexity of the graph. Density, assortativity, and reciprocity offer insights into the connectivity and interaction patterns among nodes, reflecting how edges are distributed and whether similar nodes preferentially connect to each other. Additionally, the clustering coefficient and effective diameter provide a view of the overall compactness and reachability within the graph.

Density. The density of a graph measures its compactness and connectivity. In this study, density is calculated as:

$$D = \frac{|E|}{|V|(|V|-1)}$$

where $|E|$ is the number of edges, and $|V|$ is the number of nodes. In token transfer graphs, a lower density suggests a fragmented or developing market, indicative of fewer interactions or participants. Conversely, a high density indicates a mature market with frequent transactions between participants. This distinction is crucial for understanding market dynamics.

Assortativity. The assortativity coefficient quantifies the tendency of nodes to connect with others that share similar attributes. Specifically, assortativity is calculated by:

$$r = \frac{\sum_{(i,j)\in E} \left(f(i) - f_1\right)\left(f(j) - f_2\right)}{\sqrt{\sum_{(i,j)\in E} \left(f(i) - f_1\right)^2 \sum_{(i,j)\in E} \left(f(j) - f_2\right)^2}}$$

This metric is particularly relevant in token transfer graphs, as it measures how frequently addresses transact with others of similar characteristics. A higher assortativity may indicate a market dominated by similar types of transactions or participants. However, it is important to note that this is a trend observed in our data rather than an absolute rule. Understanding this property aids in identifying market segmentation.

Reciprocity. Reciprocity measures the likelihood of directed connections being reciprocated. It is calculated by:

$$\rho = \frac{|\{(i,j) \in G : (j,i) \in G\}|}{|E(G)|}$$

This metric is crucial for understanding mutual interactions between addresses, such as reciprocal trading patterns. In token transfer graphs, a higher reciprocity suggests a strong bidirectional transactional relationship, indicating trust or partnership between nodes. This insight is vital for assessing the stability of relationships within the graph.

Clustering Coefficient. The clustering coefficient measures how closely nodes in a graph tend to cluster together. This metric is essential in token transfer graphs, as it indicates the extent to which nodes form tightly-knit groups, which may suggest collusive behavior or strong community structures. We primarily use the average clustering coefficient to assess overall network cohesion and the potential for collaborative behavior among participants. It is calculated as:

$$C_{\text{avg}} = \frac{1}{n}\sum_{i=1}^{n} C_i$$

$$C_i = \frac{2T(i)}{k_i(k_i-1)}$$

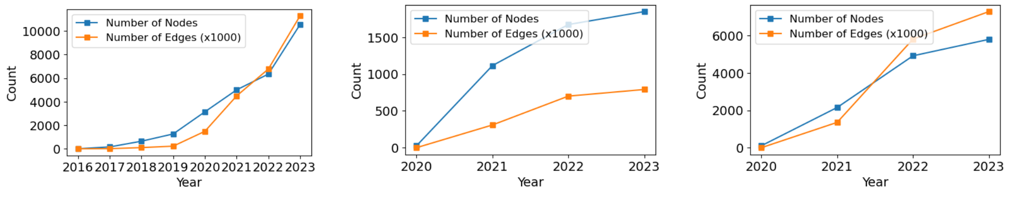
In the token transfer graph, a higher average clustering coefficient suggests a network characterized by prevalent cliques or groups that engage in frequent interactions, potentially indicating tight-knit trading communities.

Effective Diameter. The effective diameter provides insight into the average separation between node pairs across the graph. We measure the effective diameter by performing breadth-first search (BFS) from a sample of randomly selected nodes to provide a broad and representative overview of the graph's structure. The effective diameter is then defined as the 90th percentile of the shortest path lengths obtained from these BFS runs. This approach estimates how far apart nodes are on average, considering the most representative paths rather than extremes. The effective diameter reflects how easily a token can circulate within the network, a key factor in assessing liquidity and market efficiency. This metric is particularly important for understanding the graph's accessibility.

D Temporal Properties Analysis

To reveal the temporal changes in the GoG systems of the three blockchains, we analyze the yearly variation of some fundamental properties of the global graphs. Nodes represent tokens, and an edge between two nodes indicates that the tokens share common addresses during that year.

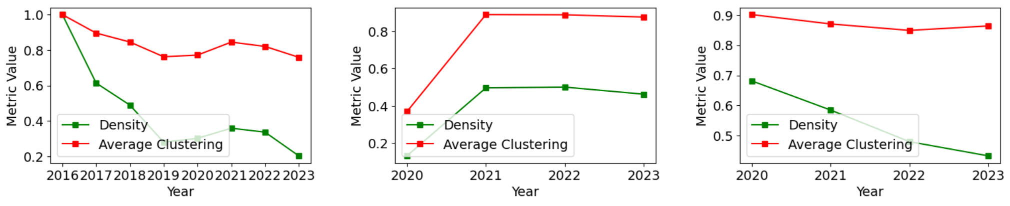
First, we examine the dynamics of the number of nodes and edges, as illustrated in Figure 6. Across the Ethereum, Polygon, and BSC ERC20 token networks, we observe a consistent trend of significant growth in both nodes and edges. This growth reflects increased adoption and diversification of blockchain platforms. Over the past three years, the average increase in the number of nodes in the global graphs is 42.49% for Ethereum, 33.08% for Polygon, and 65.18% for BSC. These figures indicate substantial changes in the dataset. Notably, Ethereum exhibits the most mature growth pattern, particularly with a significant acceleration since 2020. In contrast, Polygon shows robust growth; however, it has a slower increase in edges compared to nodes, suggesting a less interconnected network than Ethereum's GoG. Meanwhile, BSC experiences a rapid rise in both nodes and edges but begins to show signs of stabilization in 2023, indicating a maturing of its initial expansion phase. These patterns highlight that while all networks are expanding, the nature and rate of growth vary among the different blockchains.



(a) Ethereum (b) Polygon (c) BSC

Figure 6: Yearly number of nodes and edges of three global graphs.

Second, we analyze the density and average clustering coefficient of the three global graphs, as shown in Figure 7. A common trend emerges across Ethereum and BSC: both density and clustering coefficient tend to decrease as the network size increases. This trend indicates sparser connections as these networks expand, especially pronounced in the BSC network, which reflects significant diffusion from its originally dense structure. Conversely, Polygon exhibits a different pattern; both metrics initially increase and then stabilize. This indicates that the GoG not only grows but also effectively maintains or enhances its clustering. Such behavior suggests robust internal structuring that preserves community integrity even as the network scales. These observations highlight varied adaptive strategies within blockchain networks, with the Polygon GoG notably sustaining community cohesion amidst growth.



(a) Ethereum (b) Polygon (c) BSC

Figure 7: Yearly density and clustering coefficients of three global graphs.

E Model Implementation Details

In this section, we introduce the models and hyperparameters we used for the multi-class classification and anomaly detection tasks.

E.1 Multi-class classification

Models. We conduct experiments on two groups of models: (1) 5 GNN models on individual graphs, and (2) 3 GoG-based GNN models on graphs-of-graphs.

Group (1) includes GNN models for individual graphs:

- *Graph Convolutional Network (GCN)* [31] utilizes a layer-wise propagation rule based on spectral graph convolutions, enabling it to learn representations that capture graph structure and node features effectively.
- *Graph Attention Network (GAT)* [47] introduces an attention mechanism in the propagation step, allowing nodes to dynamically weigh the contributions of their neighbors.
- *Graph Isomorphism Network (GIN)* [48] is designed to capture the power of the Weisfeiler-Lehman graph isomorphism test. It approximates the Weisfeiler-Lehman graph isomorphism test by adjusting aggregators to better distinguish between different graph structures.
- *Residual Graph Convolutional Network (ResidualGCN)* [49] incorporates residual connections into the graph convolutional layers to improve gradient flow during training, which enhances the learning of deeper GNN architectures by mitigating the vanishing gradient problem.
- *GraphSAGE* [50] generates node embeddings by sampling and aggregating features from a node's local neighborhood. Its inductive learning framework supports embedding generation for unseen data, making it scalable and efficient for large graphs.

Group (2) includes models designed for graphs-of-graphs:

- *Semi-supervised Graph Classification via Cautious Iteration (SEAL)* [20] utilizes a self-attentive graph embedding method with GCN as a backbone to embed graph instances into fixed-length vectors, facilitating graph-based classification tasks. It enhances the encoding of local graph structures and their relationships within a larger graph context.
- *Graph of Graphs Neural Network (GoGNN)* [19] extends traditional GCN capabilities by integrating an attention-based pooling mechanism and GAT. It effectively identifies significant substructures within local graphs and interactions within the interaction graph, providing a powerful framework for analyzing complex graph relationships.

- *Denoising Variational Graph Autoencoder (DVGGA)* [39] employs a denoising variational autoencoder combined with a self-attentive graph neural network and a readout operation. This model is adept at handling noise in graph data, making it suitable for tasks requiring robust feature extraction and anomaly detection in noisy environments.

Model structures. For GNN models targeting individual graphs, we employ a configuration that includes two GNN layers followed by a fully connected layer for classification. This two-layer setup, consistent with the backbone design of SEAL [20], ensures fair comparisons. Each layer transforms node features to enhance feature extraction, using ReLU activation and dropout for regularization. Following the convolution layers, a global mean pooling layer aggregates node features into a cohesive graph-level representation. This representation is then processed through a fully connected layer, which outputs class probabilities using a logarithmic softmax function. For GoG models, we utilize publicly available code from the Github repositories of the original studies. For GoGNN and DVGGA, we adapt the original code from edge prediction to node classification tasks on the global graph.

Hyperparameters. For individual GNN models, we configure each layer with a dimension of 16, a dropout rate of 0, a learning rate of 0.01, and set the number of training epochs to 50. Cross-entropy serves as the loss function. For GoG-based models using a single GCN model as the backbone, we ensure that the dimensions and dropout rates are consistent with those of the individual GNN models. To fine-tune additional hyperparameters, we experiment with various settings listed in Table 6 to achieve optimal performance.

Table 6: GoG models parameter settings.

Model	Parameter	Values
SEAL	First dense neurons	16, 32, 64
	Second gcn dimensions	8, 16
	Number of epochs	50, 100, 150
	Weight	0, 0.001, 0.00001
GOGNN	Nhid	32, 64, 128
	Number of epochs	50, 100, 150
	Pooling rate	0.4, 0.5, 0.6
DVGGA	Vgae hidden dimensions	8, 16, 32
	Number of epochs	50, 100, 150

E.2 Anomaly Detection

Models. We test two groups of models: (1) 4 models for multivariate anomaly detection, and (2) 5 models for the graph anomaly detection.

Group (1) includes probabilistic-based and outlier ensembles methods designed for multivariate anomaly detection:

- *Copula-Based Outlier Detection (COPOD)* [54] is a probabilistic model that leverages the advantages of copulas for outlier detection. It does not assume a normal distribution of data, making it robust and effective in identifying outliers in various datasets with complex distributions.
- *Isolation Forest (IForest)* [55] utilizes a decision tree structure to isolate outliers by randomly selecting features and split values between the feature's maximum and minimum. Its efficiency and scalability make it well-suited for large datasets.
- *Deep Isolation Forest (DIF)* [56] extends the traditional isolation forest by incorporating deep learning techniques to enhance its capability to handle high-dimensional and complex structured data.
- *Variational Autoencoder (VAE)* [57] is a generative model that uses a neural network architecture to model data distributions and encode data into a latent space. It is widely used for anomaly detection by reconstructing inputs and measuring reconstruction errors to identify anomalies.

Group (2) includes anomaly detection methods on graphs, primarily utilizing GNN combined with Autoencoder techniques:

- *Graph Autoencoder (GAE)* [58] employs a graph convolutional network to encode the graph structure into a latent space, then reconstructs the graph to identify anomalies by measuring reconstruction loss.
- *Detection of Outliers in Network Data (DONE)* [59] integrates graph structural information with node feature information to detect anomalous nodes effectively within graph data.
- *Deep Anomaly Detection on Attributed Networks (DOMINANT)* [60] uses a deep autoencoder model adapted to graph data, enhancing the ability to capture non-linearities and complex patterns in the data, which helps in identifying both global and local anomalies in graphs.
- *Anomaly Detection with Autoencoder (AnomalyDAE)* [61] is an autoencoder-based model that particularly focuses on detecting anomalies in dynamic graphs by learning a representation that captures both the graph structure and changes over time.
- *Contrastive Learning for Anomaly Detection (CoLA)* [62] utilizes contrastive learning to differentiate between normal and abnormal nodes, leveraging the discriminative power of contrastive loss to enhance anomaly detection performance in graph settings.

Hyperparameters. We test on the following hyperparameters in Table 7 and select the best setting with superior performance.

Table 7: Models of anomaly detection parameter settings. n represents the number of features.

Model	Parameter	Values
COPOD	Contamination	0.01 to 0.1 (linear space)
Isolation Forest	Number of estimators	100, 200
	Maximum samples	256, 512
DIF	Contamination	0.01 to 0.05 (linear space)
VAE	Encoder neurons	$n/4$, $n/2$, min(20, n)
	Decoder neurons	$n/4$, $n/2$, min(20, n)
	Contamination	0.1 to 0.3 (linear space)
DOMINANT, DONE, GAE,	Hidden dimensions	16, 32, 64
AnomalyDAE, CoLA	Learning rate	0.01, 0.005, 0.1
	Number of epochs	50, 100, 150

F Global Link Prediction

Link prediction is an essential task in graph learning, widely applied in recommendation systems [74] and social media analysis [75]. In the context of blockchain analysis, predicting interactions between tokens is essential for forecasting future market behaviors. This section focuses on global edge prediction, specifically aiming to forecast interactions for newly launched tokens using information from existing tokens.

Models. We compare two groups of models based on the previous section subsection 5.1. The first group consists of traditional Graph Neural Network (GNN) models applied to global token graphs. The second group includes Graphs of Graphs (GoG) models, which leverage the hierarchical structure of token-to-token interactions. We provide a detailed comparison of performance metrics to substantiate our claims regarding the effectiveness of these models.

Settings. Our analysis focuses on the most recent tokens launched within the past year. We divided global token-token interactions into training and test sets, following an 80/20 ratio based on the tokens' launch times. Node degree serves as the primary feature for local graph embeddings, consistent with our approach in the classification task. We evaluate model performance using accuracy and AUC, supplemented by precision and recall to provide a comprehensive assessment.

Results. The performance of global edge prediction methods across three blockchains is summarized in Table 8. As shown, GoG models do not consistently outperform individual GNN models, particularly on the BSC dataset. One potential reason for these results is that the node degree, used as a node feature in this experiment, may not be as effective for predicting global edges as it is for

Table 8: Edge prediction performance by blockchain.

	Ethereum		Polygon		BSC	
Model	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
GCN	58.07±0.36	62.02±0.23	59.64±1.71	66.92±5.37	66.73±3.12	72.87±3.42
GAT	50.80±0.43	54.50±2.43	50.70±2.07	54.64±4.47	52.82±0.77	53.62±2.86
GIN	56.48±1.61	56.36±1.77	59.03±3.47	58.17±4.33	59.98±2.61	63.57±3.48
ResidualGCN	50.31±0.37	50.66±0.54	49.91±0.08	49.92±0.10	50.41±0.43	50.74±0.94
GraphSage	50.92±1.03	53.67±2.11	56.63±8.88	60.17±12.83	**71.02** ± 0.05	**78.07** ± 1.08
SEAL	57.09±1.64	64.74±4.83	56.98±4.93	64.62±10.34	56.52±4.62	58.05±6.04
GoGNN	**66.94** ± 2.08	**72.04** ± 2.41	57.10±5.21	56.72±4.75	58.99±2.77	66.25±1.84
DVGGA	50.40±1.79	62.93±1.73	**72.38** ± 1.36	**76.00** ± 0.32	63.63±4.94	69.11±3.95

classification tasks. This suggests that further exploration of edge feature engineering could enhance the predictive capabilities of GoG models for token-token interactions.

Additionally, the dynamic nature of blockchain networks presents opportunities to monitor and predict future token-token interactions, which could forecast significant market trends. However, most current GoG models are not designed with dynamic algorithms [19, 20], highlighting both challenges and potential areas for further research. We recommend future work to explore the integration of dynamic features and more sophisticated edge feature engineering to improve prediction accuracy. In summary, our findings indicate that while GoG models show promise, there is a need for further refinement and exploration of features to enhance their predictive performance in the context of blockchain networks.

G Multi-Class Graph Classification - Temporal Split

In this section, we present additional experiments that focus on predicting the class label of younger tokens using information derived from older tokens. To simulate a realistic scenario where future tokens are classified based on historical data, we implement a temporal split of the dataset. Specifically, we divide the tokens into training and test sets following an 80/20 ratio based on their first transaction timestamps. This approach enables evaluation of the model's performance within a time-sensitive context, which is crucial for applications in dynamic environments like blockchain.

The experimental settings align with those described in subsection 5.1. The results of these experiments are summarized in Table 9, which provides a comparative analysis of classification performance across different models and blockchain platforms.

Upon comparing these results with those presented in Table 4, we observe that the performance for Ethereum and BNB shows only slight differences regardless of the node-splitting method employed. However, for Polygon, we note a significant deterioration in performance. This discrepancy may be due to Polygon's status as the fastest of the major Ethereum-based chains [76], leading to varying transaction patterns across different time periods. These findings suggest that while our methods demonstrate competitive performance, further investigation is warranted to understand the underlying factors affecting classification accuracy across different blockchains.

H Graph Anomaly Detection with Deepwalk Embeddings

In this section, we present an effective method for representing token graphs in anomaly detection tasks by employing the DeepWalk algorithm [65]. DeepWalk is well-known for generating robust graph embeddings through the simulation of random walks. This approach captures the network topology and provides a nuanced representation of graph structures.

We configured DeepWalk with a walk length of 20 and performed 40 walks per node on each token transaction graph. This configuration strikes a balance between the depth and breadth of neighborhood exploration, ensuring that the embeddings accurately capture the structural and contextual nuances of the token graphs. We then aggregated these node embeddings into a unified graph-level representation by computing their mean, resulting in an embedding of 32 dimensions for each graph.

Table 9: 3-class and 5-class classification performance by blockchain (node split by time).

	Ethereum		Polygon		BSC	
Model	F1-macro	F1-micro	F1-macro	F1-micro	F1-macro	F1-micro
3-Class Classification						
GCN	60.16±5.60	87.70±0.83	22.37±0.57	48.22±0.67	50.01±5.27	57.39±3.78
GAT	57.50±6.25	87.33±1.16	26.00±2.67	48.91±1.02	51.15±6.52	59.48±5.58
GIN	60.38±5.76	87.68±0.94	21.74±1.21	48.03±0.63	42.56±2.73	56.59±3.65
ResidualGCN	40.62±8.06	83.83±1.41	22.86±1.02	48.24±0.55	48.09±5.30	60.13±2.95
GraphSage	61.71±6.27	88.25±0.97	24.91±1.87	48.72±0.55	53.86±6.99	62.16±4.28
SEAL	**67.42** ± 1.05	**88.72** ± 0.33	27.20±1.81	**49.37** ± 0.59	55.14±5.62	**64.03** ± 3.82
GoGNN	66.10±1.98	88.28±0.80	**30.85** ± 2.32	44.75±4.09	**61.80** ± 0.50	62.17±0.33
DVGGA	53.80±1.98	75.60±7.67	28.22±1.44	41.52±0.98	24.03±13.78	35.37±15.33
5-Class Classification						
GCN	38.75±5.44	85.18±0.93	12.11±0.53	41.16±0.95	26.76±3.74	47.21±4.33
GAT	37.02±5.64	85.24±1.07	**16.63** ± 3.04	42.10±2.27	28.43±4.08	49.37±5.87
GIN	22.69±1.43	80.65±0.52	12.15±0.77	41.06±0.76	22.02±2.97	43.48±5.83
ResidualGCN	41.19±5.45	85.00±1.13	12.03±0.58	41.15±0.70	24.38±4.34	47.78±6.34
GraphSage	40.51±5.82	86.31±1.10	14.97±1.69	41.98±0.69	27.89±5.48	49.06±6.83
SEAL	**48.85** ± 0.52	86.29±0.27	15.54±2.32	**42.41** ± 0.15	**30.41** ± 1.81	**52.65** ± 1.09
GoGNN	45.25±5.83	**86.36** ± 0.76	14.49±1.94	41.77±0.60	28.29±3.51	52.11±2.66
DVGGA	25.35±4.28	68.96±16.54	11.65±0.01	41.03±0.00	10.91±2.72	31.36±4.46

Table 10: Graph anomaly detection performance using DeepWalk. We report the ratio of number of non-fraud:fraud case of each data at the top.

	Ethereum (8387: 6022)		Polygon (2257: 58)		BNB (6339: 1042)	
Model	AUC	AP	AUC	AP	AUC	AP
COPOD	50.87±0.09	42.57±0.70	62.16±8.3	3.42±1.62	52.47±0.47	13.82±0.91
IForest	50.43±0.28	42.69±1.07	60.95±8.7	3.11±1.12	52.14±1.17	14.02±0.85
DIF	50.58±0.31	42.10±0.89	59.72±6.85	2.80±0.55	52.16±0.72	13.77±0.91
VAE	50.77±0.34	42.87±0.94	61.86±7.98	3.47±1.69	51.69±1.24	14.00±0.81
GAE	51.22±1.39	41.44±0.56	60.81±1.25	**5.40** ± 2.42	61.15±2.67	**24.02** ± 1.92
DONE	**68.86** ± 10.27	32.89±5.57	**71.29** ± 2.21	1.63±0.06	77.55±0.13	8.62±0.01
DOMINANT	60.92±4.57	38.12±2.63	67.15±3.41	2.43±1.00	**79.73** ± 0.07	8.42±0.01
AnomalyDAE	65.14±3.63	**46.12** ± 10.08	57.90±3.58	3.44±0.31	52.75±0.98	15.67±0.20
CoLA	50.51±0.42	41.90±0.44	59.61±3.94	2.67±0.77	54.87±0.03	14.88±0.56

The results of our anomaly detection analysis using the DeepWalk algorithm are presented in Table 10. Notably, the GoG models generally outperform multivariate outlier detection methods in our experiments, although this may vary depending on the specific characteristics of each dataset. When comparing the results in Table 7, the superiority of GoG models is evident across all three blockchains when using the DeepWalk algorithm, particularly in scenarios with high fraud rates.

It is important to note that the anomaly detection performance on Polygon remains the poorest among the chains, consistent with previous findings in subsection 5.2. While GoG models benefit from the use of the DeepWalk algorithm, the performance of multivariate outlier detection methods appears to decrease. This suggests that the DeepWalk algorithm significantly enhances the effectiveness of GoG models in identifying anomalies.

I Details of Compute Resources

We use two machine, one for experiements of inidividual GNN, one for experiements of GoG-based GNN. First, all experiments involving individual GNN models were conducted on machine outfitted with eight NVIDIA GeForce GPUs, each with a maximum power capacity of 350W and 24,576 MiB of available memory. Second, all experiments utilizing GoG-based GNN models were carried out on the machine equipped with eight NVIDIA A100-SXM4-80GB GPUs. These GPUs, each with a maximum power capacity of 400W and a substantial 81,920 MiB of memory, are specifically chosen for their high performance and large memory capacity, which are ideal for the complex and memory-intensive computations required by GoG-based GNN models.

J License

The dataset is released under the Creative Commons Attribution-NonCommercial-ShareAlike (CC BY-NC-SA) license.

K Hosting Plan

We choose GitHub as our hosting platform for both code and data due to its ease of use, cost-effectiveness, and scalability. Ensuring easy access to our data is crucial. To facilitate straightforward and reliable data retrieval, we will maintain a curated interface. We are committed to keeping our platform stable and functional, with regular updates and maintenance to ensure our repository remains up-to-date, bug-free, and efficient.

Our project is driven by a commitment to open access. By regularly updating our GitHub repository, we ensure that users have timely access to the latest data. We believe that GitHub's user-friendly environment will provide a dependable and efficient solution for sharing our data with the global community.