Fraud Detection in the Ethereum Network Using Graph Neural Networks

Sérgio Carvalhais, Tomás Gomes

[1] Centro de Ciências Tecnológicas - Universidade do Estado de Santa Catarina

ce284046@edu.udesc.br

cd638334@edu.udesc.br

Resumo. *This paper presents an approach to detect fraud in blockchain wallet networks using GNNs. The work describes the methodology based on the graph structure that blockchain data assumes, with nodes representing wallets and edges representing transactions, therefore allowing the detection of patterns in connections that characterize fraudulent activities. It takes into account some of the main types of fraud, traditional detection methods and the advantages of using GNNs, as they have the ability to capture complex relationships between transactions. We use an hybrid approach, combining GNNs with Random Forest in order to obtain the predictions. We obtained an accuracy of 88%, but a recall of 58%, which indicates that our model can be further improved, regarding the class imbalance often encountered in fraud detection datasets.*

1. Introduction

1.1. Background

Blockchain technology originally drew much attention because of its capability to ensure decentralized, safe information storage and circulation. Among the biggest blockchain platforms, Ethereum has attracted the interest of many due to the fact that it hosts so many DApps and smart contracts on its developed infrastructure. However, the rapid development in Ethereum's ecosystem has also given rise to malicious activities and fraudulent cases, which might even question the integrity of the system itself and seriously harm its users. The detection of fraudulent activities on the Ethereum network is difficult, especially considering the high volume of transactions, the complexity of blockchain interactions, and malicious actors with dynamic behavior. To address these challenges, the application of machine learning has arrived as a promising approach.

1.2. Context and Motivation

Blockchain technology, particularly Ethereum, provides a decentralized platform where transactions are verified and recorded across a distributed network of nodes.This kind of architecture provides transparency, immutability, and censorship resistance, which also make it vulnerable to fraud because these characteristics may be exploited by malicious actors to execute unauthorized transactions on the openness of the network. Anything that happens as a transaction on the Ethereum network can be viewed by all participants. Thus, it is a hot target for fraud schemes like double-spend, unauthorized transfer, and manipulation of smart contracts.

As much as Ethereum has inherent security, fraud detection remains an important challenge. Traditional techniques for fraud detection, like rule-based systems, are not enough to address the scaling and complexity aspect of blockchain transactions. Second, most of these methods are reactive rather than proactive and, hence, incapable of fraud detection in real time. Definitely, there is a demand for automated, scalable solutions which may spot fraudulent activities on time. Machine learning, especially graph-structured data-handling models, are showing promises. Since Ethereum transactions naturally form a graph, where each transaction connects accounts and accounts interact with one another, intuitively, graph-based models for fraud detection would be applicable and efficient [Zhou et al. 2018].

Considering that the nature of the Ethereum transactions is temporal, time and frequency are very important to identify suspicious behavior, and that's why it demands temporal incorporation by models. Recently, Temporal Graph Attention Networks have become a powerful tool in handling such data, where models learn not only the structure of the graph but also the temporal dynamics driving transaction patterns [Xu et al. 2019]. In this paper, we combine relational and temporal features to create a fraud detection system that would identify suspicious transactions with higher accuracy and speed.

1.3. Problem Definition

Fraud detection in the Ethereum blockchain can be considered a binary classification problem: from a set of transactions, we have to classify each one into fraudulent or not. The problem is barely classical because of the fact that the fraudulent transactions are considerably smaller by number than the legitimate transactions. Due to this, conventional models of machine learning cannot detect fraud efficiently without further techniques to handle such issues. In fact, this kind of imbalance makes it very hard for traditional machine learning models to effectively detect fraud without further techniques to address this issue [He and Garcia 2009].

In Ethereum, each transaction is represented as an edge in a graph whose nodes correspond to Ethereum accounts. The edges capture the flow of Ether between accounts, while nodes might carry information about the activities of the accounts, such as the number of transactions or the transaction volume. Both structural features in a transaction graph and characteristic features in an account involved in the transaction play an important role in fraudulent transaction detection. This gets even more complicated when temporal patterns in Ethereum transactions are considered, as fraudulent behavior may come most of the time through temporal anomalies such as bursts in transaction frequency or unusual sequences of transactions.

2. Related Work

This section reviews the existing literature on fraud detection techniques, focusing especially on machine learning approaches and graph-based methods, and will also emphasize unique challenges arising from decentralized systems.

2.1. Fraud Detection in Blockchain Systems

Fraud detection in blockchain systems has been one among the rapidly developing fields. Several works have highlighted the need for appropriate mechanisms so that fraudulent

transactions can be traced out. In relation to this, [Zhang et al. 2019] have targeted the vulnerabilities related to blockchain technology. They inferred that while blockchain provides full transparency, it has equally empowered people with the option of undertaking complex fraudulent activities such as double spending and tampering with transactions. Their work supports embedded advanced data analytics to strengthen security for these architectures.

Another relevant contribution is that by [Chen et al. 2020], which developed a hybrid model that combined anomaly detection with methods of supervised learning to classify Bitcoin fraudulent transactions. Their method was based on the use of transaction features and historical data to classify the transactions, outperforming the traditional techniques in accuracy. This research underlines very well the possibility of using machine learning techniques in fighting fraud in blockchains.

2.2. Machine Learning Approaches

The most common detection of fraud using machine learning has traditionally been done in the fields of finance and cybersecurity. Regarding the blockchain, different researchers used several methods of machine learning to detect fraudulent transactions. For instance, [Ahmed et al. 2020] used a supervised learning framework that uses decision trees and support vector machines to classify Ethereum transactions. It seems from their results that ML can learn the patterns of fraud very well, with an accuracy of more than 95%.

Deep learning methods have also recently gained popularity for fraud detection.[Wu et al. 2021] introduce a new deep neural network model, targeting the detection of fraudulent activities across Ethereum transactions. This model embeds multiple layers in order to extract hierarchical features from data in transactions, which leads to better capability for detection. This hence provides evidence that deep learning is effective in capturing those complex patterns that may be hard or even impossible for traditional machine learning methods to detect.

2.3. Graph-Based Methods

Graph-based methods have become one of the most compelling approaches for fraud detection, given the connected nature of all transactions within blockchain networks. Graphs are natural and powerful in representing relationships between participating entities - addresses and transactions - hence enabling sophisticated analyses of the underlying network structure.

One influential work in this area is the use of Node2Vec for generating node embeddings from transaction graphs. [Grover and Leskovec 2016] proposed Node2Vec, a framework for learning continuous feature representations for nodes in a graph. The method has seen applications in various domains including social networks and recommendation systems, with especially promising applications in fraud detection. For instance, the work of [Deldjoo et al. 2020] demonstrated how Node2Vec embeddings can bring more power to fraud detection models by enhancing relationship representation quality among transactions.

Complementary to Node2Vec, Graph Convolutional Networks (GCNs) have also been applied for fraud detection in blockchain systems. [Kipf and Welling 2017] proposed a GCN framework that extends the convolutional neural network paradigm to

graph-structured data. This approach has recently shown promises in capturing local neighborhood information and is considered appropriate for the identification of anomalous patterns across transaction networks.[Zhang et al. 2021] applied GCNs to detect fraudulent transactions in Ethereum, leading to massive improvements in fraud detection rates compared to traditional approaches.

2.4. Temporal Aspects of Fraud Detection

Another critical factor which influences fraud detection in blockchain systems is the temporal dimension of the transactions. Most of the time, fraudulent activities have distinct temporal patterns, and anyone should consider the time of the transaction when developing the models. Temporal features were identified as crucial to be inserted into fraud detection algorithms in several studies, such as [Xu et al. 2020], where there was developed a time-aware fraud detection framework that effectively integrated temporal features into its detection process. Their model integrates recurrent neural networks (RNNs) to learn temporal dependencies in transaction data, and as such, detect fraudulent pattern evolving over time. This indeed reflects the importance of considering the temporal dynamics of transactions with the goal of improving fraud detection capabilities.

2.5. Challenges in Blockchain Fraud Detection

It is true that, even with growth and development regarding methods for fraud detection, there still remain a number of challenges concerning blockchain systems. Probably the major challenge is that in a dataset there is a big class imbalance, with the percentage of instances of normal transactions being much higher than the small ones, which are fraudulent. Such class imbalance will result later on in biased models that cannot generalize well. Several methods have been proposed by different studies to try and alleviate this problem, such as the synthetic minority over-sampling technique (SMOTE) and cost-sensitive learning [He et al. 2019].

Another challenge is real-time detection. Since the blockchain transactions occur at a very fast pace, only real-time fraud detection capability could be helpful for minimizing potential losses. Most of these models, though efficient, may not be optimized for real-time processing. [Liu et al. 2021] propose exploration of techniques such as edge computing and distributed learning to enhance scalability and speed for fraud detection in blockchain environments.

3. Methodology

This section describes the methodology adopted in this work for the detection of fraudulent transactions on the Ethereum blockchain using machine learning techniques. The methodology discussion will be approached in terms of a few key phases that include data collection, data pre-processing, exploratory data analysis, feature engineering, model development, and evaluation.

3.1. Data Collection

The dataset used in this study was gathered from Ethereum transaction records that was designed to support research and development of fraud detection methods in cryptocurrency transactions. Each record include includes transaction hash, block height, timestamp of the transaction, sender and receiver addresses, transaction value, and error indicator (our target variable). More than 250,000 transactions are recorded in this dataset,

and this will serve as the backbone of analyzing fraudulent activities. According to [Zhang et al. 2019], large-scale datasets on blockchain significantly improve the reliability of fraud detection models using machine learning.

3.2. Data Pre-processing

First of all, it was the cleaning of the dataset, where irrelevant columns were removed and missing values were handled. The *Unnamed: 0* column was deleted because it did not add much value to the analysis. Then, the *To* column had 331 null values that also needed to be removed since it was a very small percentage of the whole dataset and didn't correlate to a great degree with the *isError* variable.

After that, filtering was done to retain in the dataset only those addresses that had a considerable level of activity, that is, between 4 and 500 transactions, grounded by [Chen et al. 2020], who emphasized the importance of focusing on active accounts to improve the detection of subtle fraudulent patterns.

3.3. Exploratory Data Analysis

Exploratory data analysis was done in order to characterize the dataset and understand how major variables were distributed. We found that the target variable was very imbalanced, where about 93.86% of transactions were tagged as non-fraudulent/legitimate, represented as 0, and 6.1% as fraudulent, represented as 1. This will be a problem during the training of any machine learning model since most algorithms easily get biased toward the majority class.

Moreover, the distribution of transaction value showed many zero-value transactions, which can be mapped to several scenarios, such as the deployment of or interactions with smart contracts, among others, or even test transactions.

Another exploratory data analysis was performed on the *Value* variable, which is, in fact, a very important variable since it explains the size and importance that transactions may have, especially when fraudulent activities are being sought.

The analysis showed that the mean value of a legitimate transaction is about 5.10 *Ether*, the median value is roughly about 0.03 *Ether*, and the standard deviation is 214.21 *Ether*. However, for fraudulent transactions, the average is roughly 0.02 *Ether*, the median value is 0.00, and the standard deviation is 0.34 *Ether*. This drastic difference suggests that the fraudulent transactions are of a relatively smaller size compared to variations in transaction sizes of normal ones.

Insights from this analysis pinpoint that any model development for fraud detection needs to bear in mind the distribution of transaction values.

3.4. Feature Engineering

Feature engineering is a significant enhancement in any machine learning model. In this study, various features were derived from the transaction data, focusing on both node features and edge features. Node features are explained in Table 1 and edge features are described in Table 2.

The features were then normalized and transformed into suitable formats for input into our model.

3.5. Model Development

To address the problem of fraud detection, a customized GNN was developed using PyTorch Geometric. The network integrates both node and edge features while employing attention mechanisms to capture the relational importance between nodes in the Ethereum transaction graph.

The architecture uses GAT, known for their strength in assigning levels of importance to neighboring nodes based on their respective relevance. This attention mechanism is especially useful in blockchain networks, as specific patterns in transactions can serve as strong indicators of fraudulent behavior. Each node is described by features that include degree metrics, transaction frequencies, and temporal properties, while edges are described by features such as transaction amount and directional ration, as explained in section 3.4. During pre-processing, the features from both nodes and edges are concatenated together so that the model will be able to use all available information.

With this being said, the architecture consists of two attention layers, each with multiple heads. These layers propagate information across the graph, refining node representations, known as embeddings, and aggregating features from neighbors in the process. Further, each of these attention layers is followed by a batch normalization layer and dropout to enhance generalization and reduce overfitting. Finally, the last layer is a linear classifier, which outputs the probability of each node being fraudulent or legitimate.

The intermediate embeddings obtained from the first attentional layer are kept and later on used as input for a downstream Random Forest (RF) classifier. Such a hybrid strategy gets the best of both worlds in terms of representation capabilities of GNNs and, at the same time, robustness from classical algorithms regarding the final classification decision [H 2022].

3.6. Model Training

In the training process of the GNN, it classifies the nodes into either fraudulent or legitimate in a supervised learning manner, where a node will be labeled depending on whether the corresponding node has participated in a fraudulent transaction. The dataset was split into 70% for training, 15% for validation, and 15% for testing, with stratification to guarantee that each subset has the same number of fraudulent nodes. The optimizer used is Adam and the loss metric is Cross-Entropy loss, which is used to minimize the error in binary classification.

Optuna, a hyperparameter search automation framework, was used for optimization of the GNN hyperparameters. This included tuning over the number of hidden channels, attention heads, and also the learning rate.

Early stopping was done to avoid over fitting, with patience set to 20 epochs. This monitored the validation loss during training and stops the process when there is no further improvement after twenty consecutive epochs elapse. It ensures that the model does not over fit the training data but generalizes well on unseen data.

In Figure 1, the loss curve demonstrates the reduction in both training and validation loss over epochs. It is expected that validation loss remains consistently higher than training loss, as the model is evaluated on unseen data. The model converges to a validation loss of approximately 0.3, which reflects its ability to generalize effectively while

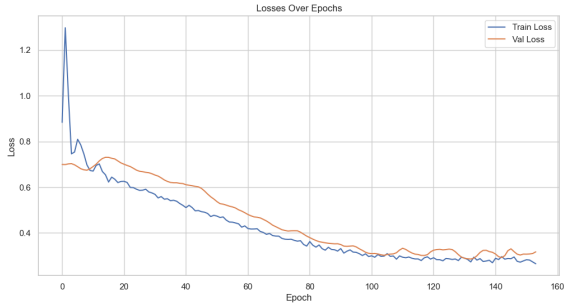

Figure 1. Train and validation loss per epoch.

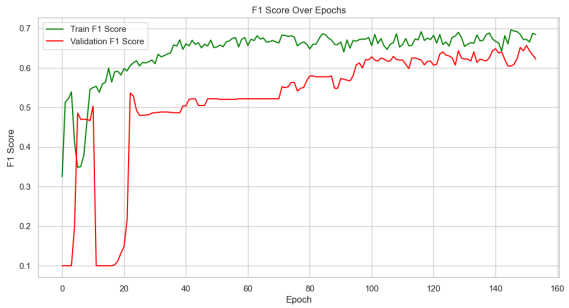

Figure 2. Train and validation F1-score per epoch.

avoiding overfitting.

Based on Figure 2, the behavior of the training and validation F1-scores with respect to epochs provides fundamental insights into the model's learning process.

Accordingly, the validation F1-score has always been lower than the training F1-score throughout training. This is expected, since the model was tested on data it had not seen before. However, this gap does not impede the growth of both scores upwards over the epochs, meaning the model learns from the data and improves step by step in classification.

We first observe large variations in the training F1-score throughout the first half of training, and then somewhat larger variations for early epochs on the validation F1-scores. This fluctuation is expected since the model will be adjusting weights and parameters. Beyond that, there's very gradual convergence and F1 scores have somewhat leveled out by the 100th epoch.

This is a sign of a healthy training process where the model generalizes well with techniques such as early stop; the ever-increasing validation F1 score shows that this model generalizes well to new, unseen data and justifies the performed hyperparameter tuning and choice of training procedure.

The node embeddings generated by the GNN were extracted after training and used as input features for a RF classifier, that was also subject to Optuna in order to optimize number of estimators and maximum depth. This way, it gave another level of abstraction to enhance the model capability of distinguishing between fraudulent versus legitimate transactions.

3.7. Evaluation

Precision, recall, F1-score, and the area under the receiver operating characteristic curve (AUC-ROC) are some of the metrics used in the performance evaluation of this model. These have been chosen to provide an all-round performance measurement for this model on fraudulent transaction detection. As datasets dealing with fraud detection are usually class-imbalanced, like ours, emphasis was directed toward recall and F1-score. Reducing false negatives was prioritized over minimizing false positives, as the consequences of failing to identify a fraudulent transaction are more severe than incorrectly flagging a legitimate transaction as fraudulent.

It should be mentioned that the GNN model was evaluated with respect to performance on a separate test set, as mentioned in 3.6. In contrast, each performance measure of the GNN-RF model was subjected to 5-fold cross-validation. Therefore, all the measures mentioned in this section are generalizable or robust.

4. Results and Discussion

4.1. Performance Evaluation

Table 3 summarizes the main performance metrics obtained from testing both the GNN with and without RF. Notably, the GNN-RF model outperforms the GNN in all evaluated metrics, demonstrating an improvement over the traditional GNN method for node classification.

The accuracy of 88% indicates that the model is effective in identifying both fraudulent and legitimate transactions. However, it's important to note that in imbalanced datasets, a high accuracy might be biased towards the majority class. Therefore, the analysis of other metrics, such as AUC-ROC and precision, provides a more balanced view of the model's performance.

Despite these high scores, the relatively low recall of 58% highlights a critical limitation: the model struggles to identify fraudulent transactions accurately. This issue is particularly significant in fraud detection, where failing to identify a fraudulent transaction can lead to substantial financial losses. Recall is a key metric, and its relatively low value emphasizes the need for improvement in this area.

The F1-Score of 60% reflects a moderate balance between precision and recall. This indicates that the model is performing reasonably well, but there is still room for improvement, especially in recall. As the model prioritizes minimizing false positives, it sacrifices some ability to correctly identify all fraudulent transactions. Further optimization could improve this balance and reduce the number of false negatives.

The nature of the problem also contributes to these challenges, as the fraudulent transactions are most of the time very similar to their legitimate counterparts. Frauds are designed to closely emulate regular patterns, which provides some inherent difficulties for machine learning models to be really able to distinguish the positive class from the negative one. This subtlety of differences enhances the issue of imbalance in datasets, whereby generalization is limited due to the rarity of fraudulent examples.

4.2. Limitations

Although the model demonstrated high accuracy and good discriminatory ability with an AUC-ROC score of 80%, a detailed analysis of the metrics revealed that the model still faces challenges, particularly regarding recall, which stood at 58%. The lower recall can be explained by the imbalanced nature of the dataset, where fraudulent transactions are much less frequent than legitimate ones. This causes the model to favor classifying most transactions as legitimate, missing out on fraudulent cases.

In practice, a fraud detection model must be highly sensitive to minimize the risk of unrecognized fraudulent transactions. While our solution proved effective, improvements in recall could make it even more applicable in real-world fraud prevention scenarios. The implementation of additional techniques, such as penalizing false negatives, can be explored to improve fraud identification.

Another limitation involves scalability and real-time detection capability for the proposed model. Most fraud detection systems have to handle a high volume of transactions in real time, which demands rapid computation of features and classification. In the case of the GNN-RF model, the computational cost associated with generating graph-based features and updating the model as new transactions and nodes are added can be substantial. This may reduce the capability of the system to offer timely and effective fraud detection in environments which have high transaction rates.

Moreover, periodic updates of the model are necessary to adapt to the evolving fraud patterns and maintain performance. However, frequent retraining and recalibration can be resource-intensive and challenging to implement in practice. For real-world deployment, it is important to design a model with a long life cycle, so that even with minimal updates, it would remain effective while maintaining computational efficiency.

The solution may be the development of incremental learning methods that would allow the model to update itself with new data without the need for full retraining. This would drastically reduce the computational load while at the same time keep the model relevant in dynamic environments. Furthermore, optimization of graph structure and the use of approximate algorithms for feature computation could help in scalability and achieve real-time processing.

5. Conclusion

This study presented a fraud detection model based on Graph Neural Networks (GNN) combined with Random Forest (RF), evaluated across several metrics. The model showed promising performance, achieving 88% accuracy and a 80% AUC-ROC. However, the 58% recall revealed significant challenges in fraud detection, which is a crucial aspect of preventing financial losses.

Our study contributes to the existing literature on fraud detection by combining deep learning techniques like GNNs with classical supervised learning methods such as Random Forests. This approach yields promising results, but also highlights the importance of focusing on metrics beyond accuracy when dealing with imbalanced classification problems.

To improve model performance, it is necessary to explore threshold adjustment techniques to improve recall. Additionally, experimenting with different GNN architec-

tures, such as GraphSAGE could help enhance the model's ability to capture complex relationships in the data. Moreover, a larger dataset could be used in order to track this model performance, as it could learn better if there was more data to train from.

With this being said, fraud detection is a critical area in various financial and commercial sectors, and that's why this study paves the way for future research that combines graph neural network techniques and supervised learning to create more accurate and robust models for fraud prevention.

References

Ahmed, E., Mahmood, A. N., and Hu, J. (2020). A survey of network anomaly detection techniques. *Computers & Security*, 102:101582.

Chen, T., Wu, J., and Zhang, Y. (2020). A hybrid model for fraud detection in bitcoin transactions. *Journal of Network and Computer Applications*, 165:102693.

Deldjoo, Y., Khoshgoftaar, T. M., and Seliem, M. (2020). A survey of graph-based fraud detection techniques. *ACM Computing Surveys*, 53(5):1–36.

Grover, A. and Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864. ACM.

H, C. (2022). How to boost your gnn. Accessed: November 22, 2024.

He, H. and Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284.

He, H., Wu, H., and Wang, J. (2019). An improved smote algorithm based on density and its application in imbalanced datasets. *Soft Computing*, 23(2):704–720.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Liu, Y., Zheng, Y., and Chen, L. (2021). Real-time fraud detection in blockchain transactions using edge computing. *IEEE Internet of Things Journal*, 8(5):3826–3835.

Wu, Z., Pan, S., and Long, G. (2021). Graph neural networks: A survey. *ACM Computing Surveys*, 54(4):1–35.

Xu, C., Liu, Y., and Zhang, S. (2020). Time-aware fraud detection in blockchain transactions. *ACM Transactions on Internet Technology*, 20(1):1–24.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019). How powerful are graph neural networks?

Zhang, Y., Li, J., and Liu, X. (2021). Graph convolutional networks for fraud detection in ethereum. *Future Generation Computer Systems*, 117:112–121.

Zhang, Y., Wang, Y., and Zhao, X. (2019). A survey of blockchain security issues and challenges. *IEEE Access*, 7:18187–18206.

Zhou, H., Zhang, J., and Zhao, Z. (2018). Graph neural networks: A review of methods and applications. *Journal of Computer Science and Technology*, 33(4):652–688.

Table 1. Node Features

Feature	Description
Total Degree	The total number of connections (edges) with other nodes, providing insight into the activity level of an address.
Out-Degree	The number of transactions sent by a node, which can indicate potential attempts to disperse assets.
In-Degree	The number of transactions received by a node, useful for identifying nodes accumulating resources.
Out-Degree Ratio	The ratio of out-degree to in-degree, indicating the behavior of a node in terms of asset dispersal.
In-Degree Ratio	The ratio of in-degree to out-degree, revealing whether a node is more of a receiver than a sender.
Sum of Transactions	The total value transacted by a node, which can highlight significant activity or potential fraud.
Transfer-Out Transaction	The sum of values sent by a node, relevant for identifying asset dispersal behaviors.
Transfer-In Transaction	The sum of values received by a node, indicating accumulation patterns.
Transaction Difference	The difference between incoming and outgoing transactions, providing insight into whether a node is an accumulator or disperser.
Transaction Ratio	The ratio of incoming to outgoing transactions, which can indicate unusual behavior.
Transfer-In Ratio	The ratio of incoming transactions to the in-degree, revealing accumulation patterns.
Transfer-Out Ratio	The ratio of outgoing transactions to the out-degree, indicating active dispersal.
Number of Neighbors	The count of directly connected nodes, useful for assessing the centrality of a node.
Inverse Timestamp Frequency	Calculated as the inverse of the average time between transactions, providing insights into transaction frequency.

Table 2. Edge Features

Feature	Description
Out-Degree	The out-degree of the source node of the edge, indicating the number of transactions it sends.
In-Degree	The in-degree of the target node of the edge, indicating the number of transactions it receives.
Out-Degree Ratio	The ratio of the out-degree of the source node to its in-degree.
In-Degree Ratio	The ratio of the in-degree of the target node to its out-degree.
Transfer-Out Transaction	The total value of transactions sent by the source node.
Transfer-In Transaction	The total value of transactions received by the target node.
Transfer-In Ratio	The ratio of the total value received by the target node to its in-degree.
Transfer-Out Ratio	The ratio of the total value sent by the source node to its out-degree.

Table 3. Performance Metrics for Fraud Detection Model

Metric	GNN-RF	GNN
Accuracy (%)	88.0	87.9
Precision (%)	67.0	57.0
Recall (%)	58.0	52.0
F1-Score (%)	60.0	51.0
AUC	80.0	79.0