BitSentry: A Graph-Based Approach for Tracking Illicit Transactions in Bitcoin

M.Tech Thesis

by

Vikas Chauhan

DEPARTMENT OF COMPUTER SCIENCE

AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY

INDORE

MAY 2025

BitSentry: A Graph-Based Approach for Tracking Illicit Transactions in Bitcoin

A THESIS

Submitted in partial fulfillment of the

requirements for the award of the degree

of

Master of Technology

by

Vikas Chauhan

2302101014

DEPARTMENT OF COMPUTER SCIENCE

AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY

INDORE

MAY 2025

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **BitSentry: A Graph-Based Approach for Tracking Illicit Transactions in Bitcoin** in the partial fulfillment of the requirements for the award of the degree of **Master of Technology** and submitted in the **Department of Computer Science and Engineering, Indian Institute of Technology Indore,** is an authentic record of my own work carried out during the period from July 2023 to May 2025 under the supervision of Dr. Chandresh Kumar Maurya, Indian Institute of Technology Indore, India.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

Vikas Chauhan

19/May/2025
Signature of the Student with Date

(Vikas Chauhan)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

CKmaurya 19/5/2025
Signature of Thesis Supervisor with Date

(Dr. Chandresh Kumar Maurya)

Vikas Chauhan has successfully given his M.Tech. Oral Examination held on **30/April/2025**.

CKmaurya
Signature(s) of Supervisor(s) of M.Tech. thesis

Date: 20/5/2025

Subhra Mazumdar 20.05.2025
Signature of Chairman, PG Oral Board

Date:

Signature of HoD

Date: 20-May-2025

ACKNOWLEDGMENT

I would like to take this opportunity to express my sincere gratitude to all those who supported me throughout the course of this research. Their contributions, whether academic, technical, or personal, made this journey both insightful and rewarding.

First and foremost, I would like to extend my heartfelt thanks to my supervisor, **Dr. Chandresh Kumar Maurya**, for his constant guidance, encouragement, and support. His insightful feedback and research direction were instrumental in shaping this work. I am also thankful for his patience and dedication in mentoring me at every critical step of the research process.

I am equally grateful to my co-supervisor, **Dr. Subhra Mazumdar**, for her valuable suggestions and support throughout this project. Her perspective and timely advice added great value to the work and helped me stay focused on the research goals.

I would also like to acknowledge the support provided by **Sonu Kumar** and **Gaurav Patel**, whose assistance with the DGX server and technical infrastructure was essential for the successful execution of the experiments in this thesis.

I am thankful to all my friends and colleagues who directly or indirectly encouraged and motivated me during this period. Their camaraderie made this experience more enjoyable and less stressful.

Lastly, I would like to thank my family for their unconditional love and support, which gave me the strength and motivation to pursue my academic journey.

This work would not have been possible without the collective contributions of all these individuals. I remain deeply grateful.

Dedicated to My Family

ABSTRACT

Detecting illicit transactions in financial networks is crucial to modern anti-money laundering (AML) efforts, particularly within decentralized systems like Bitcoin. Inherent anonymity and obfuscation strategies make tracing unlawful activities significantly more challenging. Traditional rule-based detection mechanisms and standard machine learning techniques often fail to generalize well against evolving patterns of money laundering, which are deliberately engineered to circumvent known safeguards. Recent advances in Graph Neural Networks (GNNs) have shown substantial promise in fraud detection, given their ability to capture structural and relational intricacies in transaction networks. Through graph restructuring, frameworks such as FraudLens [ACSAC 23] have addressed issues like label imbalance and topological bias. However, these approaches often overlook deeper graph-theoretic insights that can further refine the quality of input graphs before GNN processing.

This thesis proposes a novel centrality-aware transaction classification framework that enhances illicit transaction detection by integrating fundamental graph centrality measures specifically, betweenness centrality and eigenvector centrality into the graph refinement pipeline. These centrality metrics identify key nodes and transaction paths, thereby informing edge selection to emphasize meaningful relationships while reducing noise. Betweenness centrality highlights intermediary nodes frequently appearing on transaction paths and may function as mixers. In contrast, eigenvector centrality surfaces influential nodes strongly connected within the network, often associated with core members of fraudulent rings.

We also tried a feature selection mechanism using Random Forests, where only the most informative node features are retained and passed to the GNN. This step effectively reduces feature noise and dimensionality, ensuring that the subsequent Graph Convolutional Network (GCNs) focuses on high-impact attributes instead of being overwhelmed by irrelevant data. GCNs operate directly on graph structures and update each node's representation by aggregating features from its neighbors. This allows them to capture local and structural patterns indicating suspicious behavior, such as

close associations with known illicit entities or repeated involvement in circular transaction paths.

In addition, we explore the application of a Graph Transformer architecture, referred to as Graph Transformer, which, to our knowledge, has not been previously applied to illicit transaction detection in cryptocurrency networks. This model incorporates attention mechanisms to dynamically prioritize connections and nodes, offering a more flexible and expressive approach to modeling complex transactional relationships.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cryptocurrencies, particularly Bitcoin, have become a prominent part of the global financial system. As decentralized digital assets, they offer several advantages, such as peer-to-peer transactions without intermediaries, fast transaction processing, and relatively low transaction costs. These features have led to widespread adoption in various sectors, including finance, e-commerce, and even remittances. However, despite these benefits, the rise of cryptocurrencies has also brought about significant concerns, particularly for illicit activities such as money laundering, fraud, and ransomware attacks. The pseudonymous nature of Bitcoin transactions allows users to engage in these activities while masking their identities, making it difficult for traditional regulatory and surveillance systems to track and prevent such behaviours.

Bitcoin transactions are recorded on the blockchain, a distributed ledger that records every transaction in a decentralized manner. While blockchain provides transparency, it also creates a challenge for monitoring and detecting illegal activities due to the sheer volume and complexity of the data. Traditional fraud detection methods, which are based on centralized databases and manual inspection, are not scalable enough to handle the massive transaction volumes and intricate relationships in

blockchain networks. Therefore, more sophisticated techniques are needed to identify patterns indicative of illicit behaviour in the blockchain network.

One promising approach is to treat Bitcoin transaction data as a graph, where transaction IDs are represented as nodes and the flow of Bitcoin is represented as edges. This graph structure inherently captures the relationships between participants in the network and can be used to identify suspicious or fraudulent behavior based on the graph structure. A particularly effective class of techniques for analyzing graph-structured data is Graph Neural Networks (GNNs), which use message-passing mechanisms to aggregate information from neighbouring nodes. These methods effectively capture node features and relationships between nodes, making them suitable for detecting fraud in transaction networks.

This thesis explores how centrality-based graph augmentation, specifically using betweenness centrality and eigenvector centrality, can enhance the performance of Graph Convolutional Networks (GCNs) for illicit transaction detection in Bitcoin networks. By integrating centrality measures, which capture the relative importance of nodes within a graph, the thesis investigates whether GCNs can better leverage the structural properties of the transaction network to identify illicit activities. In addition to this, a Random Forest-based feature selection method is introduced to identify the most discriminative transaction features, which are then processed using a Convolutional Neural Network (CNN) to learn localized feature patterns, thereby improving classification performance on tabular representations. Furthermore, the thesis explores the use of a Graph Transformer architecture, which employs attention mechanisms to capture long-range dependencies and temporal dynamics in the transaction graph. The effectiveness of the proposed approaches is evaluated on the publicly available Elliptic++ dataset, and results are compared against baseline methods such

as GraphSAGE and unaugmented GCN models.

1.1 Motivation

The primary motivation for this research stems from the increasing prevalence of illicit activities in cryptocurrency networks. Cryptocurrencies have attracted a variety of malicious actors, including money launderers, fraudsters, and hackers, who exploit the anonymity and decentralized nature of the blockchain to carry out illegal activities. According to reports from organizations like the Financial Action Task Force (FATF) and the European Union Agency for Law Enforcement Cooperation (Europol), cryptocurrencies have increasingly facilitated illegal activities, such as ransomware attacks and darknet transactions. As cryptocurrency usage grows, so does the complexity of detecting such activities.

While traditional fraud detection techniques rely on centralized databases, pattern matching, and heuristic rules, these methods often fail to capture blockchain networks' dynamic and decentralized structure. Furthermore, these methods typically do not scale well with the size of blockchain networks, which are continually expanding. Thus, the need for scalable, efficient, and accurate methods to detect illicit activities in cryptocurrencies is crucial.

Graph-based machine learning offers a promising solution to this problem. In particular, Graph Neural Networks (GNNs) have emerged as a powerful tool for analyzing graph-structured data. In a transaction graph, the connections between wallet addresses can reveal crucial patterns that are difficult to detect through traditional methods. Centrality measures, such as betweenness and eigenvector centrality, provide a way to identify key nodes in the graph, which could potentially be involved in fraudulent activities. By using these centrality measures as graph augmentation

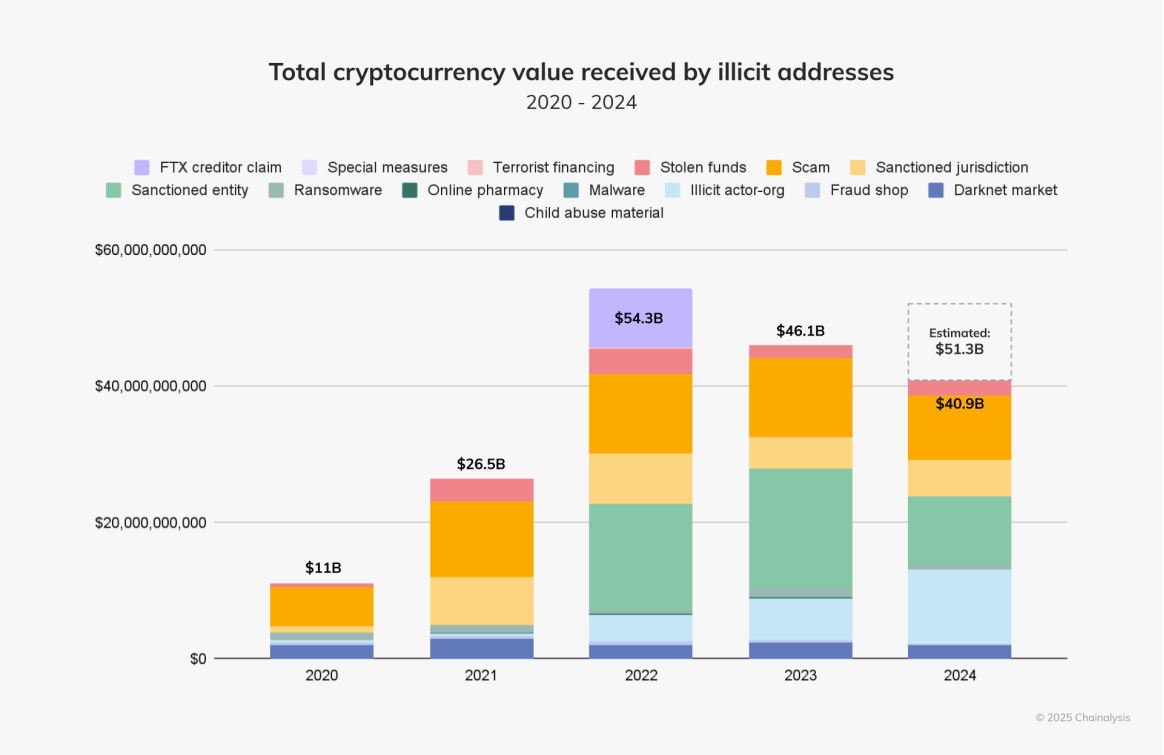


Figure 1.1: Feature selection pipeline using Random Forest and `scikit-learn` before input to GCN.

techniques, this research aims to improve the performance of Graph Convolutional Networks (GCNs) for illicit transaction detection.

Moreover, using centrality measures enables a more nuanced representation of the transaction network. Betweenness centrality identifies nodes that act as intermediaries between other nodes, while eigenvector centrality measures a node's influence in the network. These measures can reveal influential wallet addresses that may be involved in illegal activities, even when their direct transaction patterns are not overtly suspicious. By augmenting the transaction graph with these centrality-based edges, the model may be better equipped to identify complex fraud patterns that would otherwise remain undetected.

In addition to graph-based approaches, this research also incorporates a hybrid

pipeline that leverages Random Forest-based feature selection to rank the most relevant features from the raw transaction data. This filtered set of features is then passed to a Convolutional Neural Network (CNN), which is adept at identifying localized patterns and correlations among features. This method complements the graph-based strategy by enhancing the detection performance on the tabular feature space and provides an interpretable foundation through feature importance scores.

Furthermore, the study explores the use of Graph Transformer Networks, which extend the capabilities of traditional GNNs by incorporating self-attention mechanisms. Unlike GCNs, which aggregate information from immediate neighbours, Graph Transformers can model long-range dependencies, temporal patterns, and multi-hop relationships between transactions. This enables the model to detect sophisticated laundering techniques that span multiple steps and timeframes, which are often missed by simpler graph models.

Thus, the research aims to develop a comprehensive framework for illicit transaction detection by integrating centrality-based graph augmentation, feature-driven CNN pipelines, and transformer-based graph modeling. By leveraging these complementary methods, the study contributes to the creation of a robust, scalable, and interpretable solution for detecting illicit activities in cryptocurrency networks.

1.1.1 Techniques Used to hide address

Despite the pseudonymous nature of cryptocurrencies, all transactions are permanently recorded on public blockchains, making them potentially traceable. Malicious actors employ several sophisticated techniques to circumvent detection and obscure the flow of illicit funds. The most prominent are coin mixing, CoinJoin, tumbling, and peeling transactions, each designed to complicate forensic analysis and anti-money

laundering (AML) efforts.

- **Coin Mixing** Coin mixing services are widely used by criminals to obscure the origin and destination of illicit funds. In this process, multiple users send their coins to a centralized mixing service, which pools and fragments the funds, shuffles them with coins from other users, and then redistributes the mixed coins to new addresses specified by the users. This severing of the direct link between input and output addresses makes it highly challenging for investigators to trace the transaction path. Mixing services have been repeatedly linked to money laundering and have been the subject of law enforcement actions (e.g., the shutdown of Bestmixer).

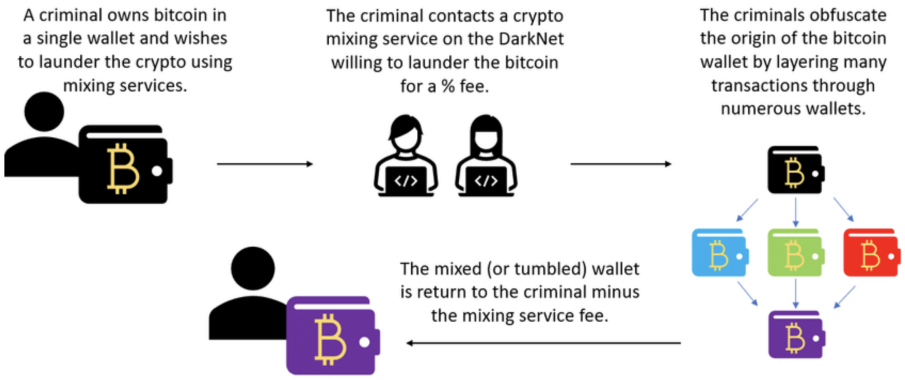


Figure 1.2: Coin Mixing method in Bitcoin network.

- **Tumbling** Tumbling services, also known as Bitcoin tumblers, are centralized platforms that automate the mixing of coins for a fee. Those seeking to launder proceeds from illegal activities often use these services. Tumblers split deposited funds into smaller amounts, introduce random delays, and send them through intermediate wallets before releasing them to the user's desired address. This process is designed to break the traceable link between the source and destination of the funds. Many tumbling services have been implicated in large-scale money

laundering operations and have faced regulatory crackdowns.

- **CoinJoin** CoinJoin is a decentralized transaction protocol that allows multiple users to combine their payments into a single transaction, thereby hiding the relationship between senders and receivers. Criminals exploit CoinJoin to make it nearly impossible for observers to determine which input corresponds to which output, as all participants' funds are aggregated and then distributed to new addresses. This technique is especially attractive for illicit actors because it does not require trust in a third party and is supported by several privacy-focused wallets.

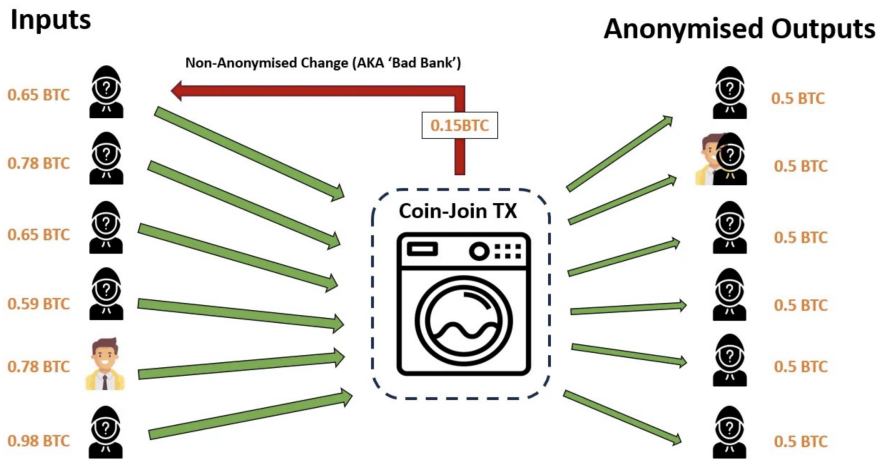


Figure 1.3: Coin join method in Bitcoin network.

- **Peeling Transactions** Peeling chains are commonly used to launder large sums by breaking them into many small, incremental transactions. In a peeling scheme, a wallet holding significant cryptocurrency sends a small portion to an exchange or another address. In contrast, the remainder is sent to a new wallet. This process is repeated, creating a long chain of transactions that gradually disperses the illicit funds. The resulting "diminishing UTXO" trail makes

it difficult for investigators to follow the money flow, mainly when automated scripts are used to execute thousands of such transactions.

1.1.2 Implications on AML

These obfuscation techniques pose significant challenges for AML systems and regulatory compliance. By masking the true origin and destination of funds, criminals can exploit the anonymity of blockchain networks to launder proceeds from cybercrime, ransomware, fraud, and other illicit activities. As a result, advanced detection methods, such as graph-based machine learning models that analyze transaction patterns and temporal relationships, are critical for identifying and disrupting these sophisticated laundering schemes.

1.2 Objectives

This thesis has the following key objectives:

Feature Selection and Local Feature Learning using Random Forest: In this objective, we focus on the dimensionality reduction and feature prioritization aspects of the dataset using a Random Forest model. The selected features—those with the highest predictive value—are then fed into a Convolutional Neural Network (CNN), which excels at learning local patterns and hierarchical representations. The goal is to reduce noise and computational complexity while enhancing the model's ability to detect suspicious transaction behaviours through spatial pattern recognition within the selected attributes.

Capturing Multi-Hop and Temporal Dependencies via Graph Transformer Networks: This objective explores the application of Graph Transformer

Networks, which combine the strengths of attention mechanisms and graph learning. Unlike traditional GNNs, Graph Transformers can effectively capture global structural dependencies, long-range interactions, and temporal dynamics in a transaction network. This is particularly beneficial for identifying sophisticated fraudulent behaviors that span multiple hops and evolve over time—patterns that are often missed by local or shallow models.

Investigate Centrality-Based Graph Augmentation: The last objective is to explore how centrality measures, such as betweenness and eigenvector centrality, can be integrated into the transaction graph to enhance the detection of illicit transactions. We hypothesize that augmenting the graph with these centrality-based edges will improve the model's ability to classify illegal wallets' addresses accurately.

Develop a Machine Learning Pipeline for Illicit Transaction Detection: We aim to design a machine learning pipeline for classifying Bitcoin wallet addresses as illicit, licit, or unknown. This will involve creating and preprocessing the graph data, training a Graph Convolutional Network (GCN) model, and evaluating its performance. The pipeline will also explore how GCNs can effectively use centrality measures to capture the underlying relationships in the transaction graph.

Evaluate Model Performance: The performance of the proposed approach will be evaluated by comparing it with existing state-of-the-art methods, such as GraphSAGE. Evaluation will be based on several classification metrics, including precision, recall, accuracy, and F1-score, focusing on how centrality-based graph augmentation improves illicit transaction detection.

Examine the Impact of Tolerance Values in Centrality-Based Augmentation: Another essential objective is to study the effect of different tolerance values in the computation of centrality measures. This will involve analyzing how varying the

tolerance thresholds for adding edges based on centrality impacts the graph's structure and, in turn, the model's performance.

By achieving these objectives, this thesis introduces three novel approaches for addressing the problem of illicit transaction detection in cryptocurrency networks. Through comprehensive experimentation and comparative analysis, the results demonstrate that the centrality-based graph augmentation approach consistently outperforms other methods. Ultimately, this work contributes to advancing the field of cryptocurrency fraud detection and highlights how incorporating centrality measures into graph structures can significantly enhance the performance of machine learning models in identifying illicit activities on the blockchain.

1.3 Technical Specifications

The experiments conducted in this thesis were designed to ensure a rigorous analysis. Below are the detailed specifications of the experimental setup:

- **Dataset:** The experiments use the Elliptic++ dataset, a publicly available dataset of Bitcoin transactions. This dataset contains over 822,000 wallet addresses and more than 1.27 million transactions, spanning several months of transaction activity. The task is to classify wallet addresses into three categories: illicit, licit, and unknown. The dataset is divided into different time steps, and the training, validation, and test sets are formed based on these time steps.

- **Graph Neural Networks (GNNs):** The model architecture used in this thesis is the Graph Convolutional Network (GCN), which is well-suited for graph-structured data. The GCN aggregates features from neighbouring nodes to gen-

erate a node representation that captures both local and global information. GraphSAGE was used as a baseline method for comparison, which aggregates information from fixed-size neighbourhoods to learn node embeddings.

- **Graph Augmentation:** Centrality measures (betweenness and eigenvector centrality) are computed using the NetworkX library. These centrality measures help add additional edges between highly central nodes, improving the model's understanding of node importance. The centrality-based edges are added with varying tolerance values, and the resulting augmented graph is used for training the GCN model.

- **Training Configuration:** The GCN models were trained for 1000 epochs. The dataset was split into training, validation, and test sets, with training occurring on time steps 7-11, validation on time steps 12-17, and testing on time steps 18 and remaining.

Chapter 2

Related work

A significant body of research has been devoted to analyzing Bitcoin and other cryptocurrency networks, with particular attention paid to trading behaviors, ownership structures, coin distributions, and transaction mechanisms. Researchers have harnessed graph-theoretic and network analysis techniques by modeling the interactions among users, wallets, and transactions as graph structures to uncover meaningful patterns, detect communities, and identify anomalous behaviors indicative of fraud or money laundering. Blockchain transactions' transparent and traceable nature makes these networks particularly amenable to such analysis. Readers may consult recent surveys and systematic reviews on cryptocurrency transaction analytics and illicit activity detection for comprehensive overviews of the field.

2.1 Traditional Machine Learning Approaches

Early work in illicit transaction detection typically relied on *traditional machine learning (ML)* models trained on carefully engineered features. Commonly used classifiers include *Random Forests*, *Multi-Layer Perceptrons (MLPs)*, *XGBoost*, *Logistic Regression*, *AdaBoost*, *Gradient Boosting Decision Trees (GBDTs)*, and *k-Nearest*

Neighbors (k-NN) [3, 2, 18, 15]. These models have shown decent performance when applied to datasets such as the *Elliptic dataset* introduced by Weber et al. [19], which provides an anonymized, temporally-ordered graph of Bitcoin transactions labeled as licit, illicit, or unknown.

The features fed into these models often include transaction-level statistics such as in-degree/out-degree, total value transferred, time since last transaction, and wallet balance. However, despite their empirical effectiveness, these models typically treat each transaction independently, ignoring the *topological structure*, *temporal evolution*, and *inter-transaction dependencies* present in real-world cryptocurrency transaction graphs. This independence assumption leads to models that are less robust to unseen or evolving laundering strategies and lack generalizability across networks or time frames.

2.2 Data Imbalance and Mitigation Strategies

A major challenge in illicit transaction detection is the *highly imbalanced class distribution*: only a small fraction of transactions are involved in money laundering or other illicit activities. To counteract this, researchers have explored a variety of *data balancing techniques*, including [18, 13]:

- **Undersampling** the majority (licit) class, which reduces training bias but discards potentially useful data.
- **Oversampling** the minority (illicit) class using duplication or synthetic instance generation (e.g., SMOTE).
- **Generative modeling** approaches, such as GAN-based frameworks or variational autoencoders, to simulate realistic illicit transactions and enrich the mi-

nority class [1].

While these techniques help mitigate imbalance-related performance degradation, they can inadvertently introduce bias or artifacts [4], potentially causing models to overfit synthetic behaviors or misrepresent real-world laundering tactics.

2.3 Rise of Graph Neural Networks (GNNs)

More recently, *Graph Neural Networks (GNNs)* have emerged as powerful tools for learning over graph-structured data, offering the ability to incorporate both *node features* and *graph topology* into a unified learning framework. Given their natural fit for blockchain transaction networks, GNNs have gained significant traction in fraud detection and anti-money laundering (AML) tasks.

Graph Convolutional Networks (GCNs), in particular, have been widely adopted to model structural relationships among transactions by aggregating information from a node's neighbors through layers of localized convolution operations [22]. Enhancements to GCNs using *attention mechanisms*, *residual connections*, *adaptive neighborhood sampling*, and *customized message-passing strategies* have further improved their ability to capture complex patterns of illicit behavior.

To address temporal aspects, researchers have extended GNN architectures with *Recurrent Neural Networks (RNNs)* [5] or *Long Short-Term Memory (LSTM)* networks [8], which help capture sequential transaction behaviors and evolution over time. These hybrid models aim to jointly model structural and temporal dynamics, crucial for detecting laundering strategies like layering, chaining, or tumbling.

The seminal work by Weber et al. [19] was among the first to use the *Elliptic dataset* for GNN-based transaction classification. They demonstrated that GNNs outperform

traditional ML models on this dataset, paving the way for a wave of subsequent work that further refined these architectures and applied them to more complex and larger-scale transaction graphs.

2.4 Graph Preprocessing for Improved GNN Performance

Beyond architectural innovations, another line of research explores *graph preprocessing techniques* to enhance downstream GNN performance without modifying the core model architecture. These strategies aim to transform or augment the input graph to amplify relevant patterns or reduce noise, making it easier for GNNs to learn useful representations.

FraudLens [11] represents a key contribution in this direction, proposing two preprocessing methods:

- **Edge based on Affinity (EA)**: Adds new edges to the transaction graph based on *Personalized PageRank (PPR)* scores. The goal is to improve connectivity among semantically related nodes, even if they are not directly connected in the raw transaction data.

- **Edge based on node features (ENF)**: Prunes potentially noisy edges by evaluating *node feature similarity*, removing connections between nodes that are deemed semantically dissimilar.

While these techniques enhance GNN performance in some settings, they have limitations. By adding connections based on a global similarity metric, EA may introduce *irrelevant or misleading edges* that dilute critical local patterns indicative of fraud. For

instance, excessive augmentation may mask chain-like transaction flows, characteristic of layering in money laundering. ENF, on the other hand, risks *removing structurally important edges* that are not reflected in feature similarity alone, such as connections involving mixer nodes or aggregators, which are often obfuscated intentionally.

Moreover, both techniques are *domain-agnostic* because they are not tailored to the specific behavioral patterns of financial crime. They do not explicitly account for known laundering typologies, such as *hub-and-spoke* models (e.g., central mixers distributing funds) or *circular transaction loops* used to disguise the origin of funds.

Chapter 3

Dataset

The dataset used in this thesis plays a crucial role in evaluating the proposed approach for detecting illicit transactions in Bitcoin networks. This study uses the Elliptic++ dataset, which provides a comprehensive collection of Bitcoin transaction data. This dataset is explicitly designed for illicit transaction detection and offers a rich set of features that can be used to model the transaction graph and train machine learning models. Below, we describe the dataset in detail, including its structure, key features, and preprocessing steps.

3.1 Dataset Description

The Elliptic++ dataset is a transaction dataset containing over 822,000 wallet addresses and more than 1.27 million transactions. The dataset spans multiple months of Bitcoin transaction activity, and each transaction includes detailed information about the sender and receiver wallet addresses and features the amount of Bitcoin transferred. The primary goal of the dataset is to classify into three categories: illicit, licit, and unknown. These classifications are based on whether the wallet is involved in activities such as money laundering, fraud, or other illegal transactions.

Properties:	**Count**
# Nodes (transactions)	203,769
# Edges (money flow)	234,355
# Time steps	49
# Illicit (class-1)	4,545
# Licit (class-2)	42,019
# Unknown (class-3)	157,205
# Features	183

Table 3.1: Statistics of the Elliptic++ Transactions Dataset

Properties:	**Count**
# Wallet addresses	822,942
# Nodes (temporal interactions)	1,268,260
# Edges (addr-addr)	2,868,964
# Edges (addr-tx-addr)	1,314,241
# Time steps	49
# Illicit (class-1)	14,266
# Licit (class-2)	251,088
# Unknown (class-3)	557,588
# Features	56

Table 3.2: Statistics of the Elliptic++ Actors (Wallet Addresses) Dataset

The dataset consists of two main components:

- **Transactions:** This component includes all the transaction records, where each transaction is represented by a pair of wallet addresses (sender and receiver) along with the transaction amount and timestamp. These transactions form the edges of the transaction graph, where each wallet address is a node.

- **Wallet Labels:** Each wallet address in the dataset is labeled as either illicit,

licit, or unknown. The illicit labels are assigned to wallets involved in fraudulent activities such as money laundering or scam operations. The licit labels are given to wallets that are involved in legitimate transactions. Unknown labels are used for wallets where the classification is not available or for new, unclassified wallets.

3.2 Key Features

The Elliptic++ dataset provides several key features that can be used to train a machine learning model for illicit transaction detection. These features are categorized into the following:

3.2.1 Transaction Features:

- **Transaction ID:** A unique identifier for each transaction.
- **Timestamp:** The time at which the transaction occurred. This feature helps track the sequence of transactions and their temporal relationships.
- **Amount:** The amount of Bitcoin transferred in the transaction. This can provide insight into the scale of transactions between wallet addresses.

3.2.2 Wallet Features:

- **Balance:** The total amount of Bitcoin held in a wallet.
- **Transaction Frequency:** The frequency with which a wallet participates in transactions. Wallets with abnormal activity, such as frequent small transfers, may indicate illicit activity.
- **Transaction Volume:** The total volume of Bitcoin transferred by a wallet.

3.3 Dataset Preprocessing

Before using the dataset for training and evaluation, several preprocessing steps are required to transform the data into a format suitable for graph-based analysis:

- **Graph Construction:** The first step in preprocessing is to represent the transactions as a graph. In this graph, each transaction ID is a node, and each money flow is an edge between two transactions. The graph is directed, with edges from the first transaction to the following. The weight of each edge can be defined as the transaction amount, providing additional information to the model.

- **Feature Extraction:** For each transaction ID, relevant features are extracted, including transaction volume and transaction frequency. The centrality measures (such as betweenness and eigenvector centrality) are calculated using the NetworkX library, which efficiently implements various graph algorithms.

- **Labeling:** transaction IDs are labeled as illicit, licit, or unknown based on the ground truth provided in the dataset. These labels will serve as the target variables during model training and evaluation.

- **Time Step Division:** The dataset is divided into discrete time steps to facilitate the modeling of temporal dynamics. For example, the dataset can be split into several time windows, such as time steps 7–11 for training, time steps 12–17 for validation, and time steps 18 and remaining for testing. This helps simulate the real-world scenario where models must predict future illicit transactions based on historical data.

- **Handling Missing Data:** A few datasets contain missing or incomplete information. These data are not much, and since there are financial transactions, and

they are connected, we can not give them any random or average value. Since this data is very low, we have removed it from our dataset.

3.4 Dataset Split

To ensure the validity and generalizability of the results, the dataset is split into three parts:

- **Training Set:** The training set is used to train the model and typically consists of data from steps 7 to 11. These time steps allow the model to learn from past transaction patterns.

- **Validation Set:** The validation set is used for hyperparameter tuning and model selection. It includes data from time steps 12 to 17. This set allows for fine-tuning model parameters without overfitting the training data.

- **Test Set:** The test set evaluates the model's performance on unseen data. It contains data from time steps 18 to 49. This set helps assess how well the model generalizes to future transaction data.

3.5 Dataset Availability

The Elliptic++ dataset is publicly available for academic and research purposes. It has been widely used in cryptocurrency fraud detection research, providing a valuable benchmark for testing and comparing detection methods. Researchers and practitioners can use this dataset to develop and evaluate machine learning models to detect illicit behavior in Bitcoin networks.

In summary, the Elliptic++ dataset provides a rich and complex transaction data source, offering numerous features that can be leveraged for fraud detection. By representing transactions as a graph and using machine learning models like GCNs, this dataset can be the foundation for developing scalable and efficient methods to detect illicit transactions in cryptocurrency networks. The Elliptic++ dataset is publicly available at https://github.com/git-disl/EllipticPlusPlus.

Chapter 4

Random Forest-Based Feature Selection

In graph-based machine learning, particularly with Graph Convolutional Networks (GCNs), the quality of input node features plays a pivotal role in the final classification performance. Feeding all available features into a GCN may introduce noise, increase computational overhead, and dilute the learning capacity of the network. To address this, we propose a feature selection pipeline that leverages the Random Forest algorithm to identify and retain only the most informative features before inputting them into the GCN model.

4.1 Methodology Overview

As illustrated in Figure 5.1, our pipeline consists of three main stages:

1. **Random Forest Feature Ranking**: A Random Forest classifier is trained using the complete set of node features and class labels. The `feature_importances_` attribute from `scikit-learn` ranks the features based on their contribution to classification performance.

2. **Selection of Top Features**: We select the top-K features (as determined empirically or through cross-validation) that contribute most significantly to model accuracy. This step ensures that irrelevant or redundant features are excluded.

3. **GCN with Selected Features**: The GCN is then trained using only the selected subset of features, resulting in a leaner and more focused input representation.

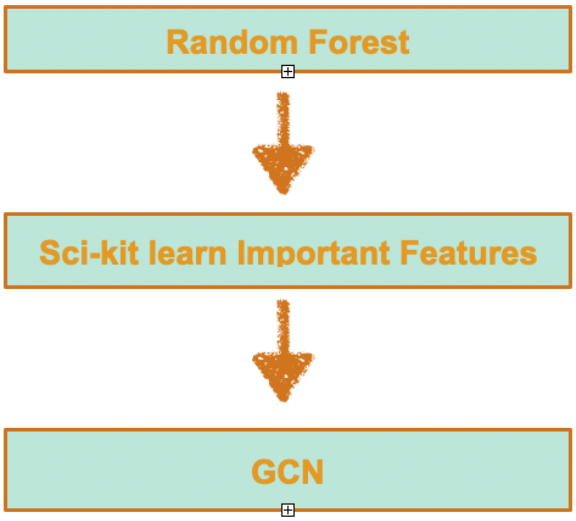


Figure 4.1: Feature selection pipeline using Random Forest and `scikit-learn` before input to GCN.

4.2 Feature Importance Analysis

Figure 4.2 presents the ranked importance of features derived from the Random Forest model. It is evident that certain local features—such as `Local_feature_48`, `Local_feature_42`, and `Local_feature_67`—exhibit significantly higher importance compared to others. These features likely capture transactional patterns indicative of illicit activity, such as abrupt changes in flow, repeated intermediary behavior, or unusual statistical outliers.

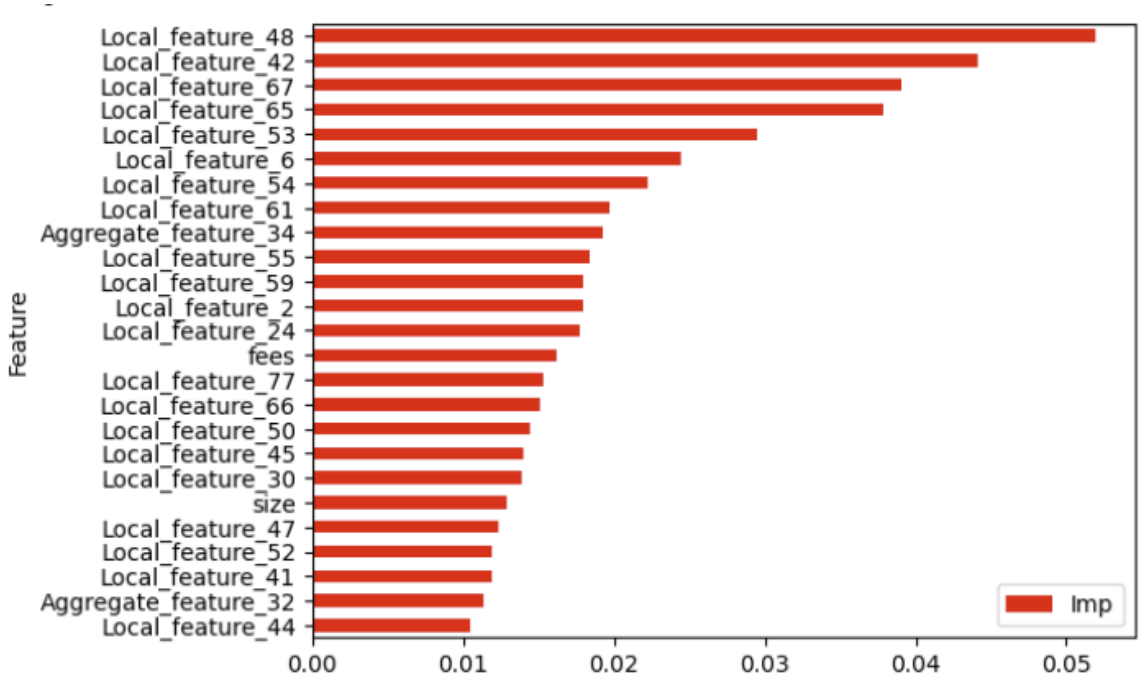


Figure 4.2: Ranked feature importances based on Random Forest.

4.3 Performance Comparison

The effectiveness of this feature selection approach is quantitatively validated through performance metrics as shown in Table **??**. When the GCN is trained on all available features, it achieves an accuracy of 77.50% and an F1-score of 74.18. In contrast, when only the Random Forest-selected features are used, the GCN's performance improves across all metrics—accuracy rises to 79.56%, precision increases to 76.03%, and the F1-score improves to 75.66.

Table 4.1: Performance of GCN with all features vs RandomForest selected features

Model	Features	Accuracy	Precision	F1-Score
GCN	All	77.50	74.20	74.18
GCN	RandomForest Selected	79.56	76.03	75.66

4.4 Interpretation and Benefits

This improvement demonstrates that intelligently selecting relevant features not only enhances classification outcomes but also reduces training complexity. The Random Forest acts as a filter that removes non-discriminative or redundant features, allowing the GCN to concentrate on the most meaningful signals in the data. This is particularly beneficial in transaction networks, where feature redundancy is common and data imbalance can skew learning.

Furthermore, this approach is model-agnostic and can be integrated with other graph learning methods such as GAT, GraphSAGE, or Transformer-based models, paving the way for more interpretable and efficient graph analytics in financial crime detection.

Chapter 5

Graph Transformer and Graph Transformer with GPE

5.1 Introduction

Graph-based neural networks have emerged as a powerful class of models for learning on structured data, particularly in detecting anomalous patterns within transaction networks. This chapter explores two transformer-inspired graph models:

1. **Graph Transformer**: A transformer-based architecture applied to graph data.
2. **Graph Transformer with GPE (Graph Positional Encoding)**: An extension of the Graph Transformer that incorporates spectral graph positional encoding to enrich node representations.

5.2 Potential of Graph Transformer Models

Graph Transformers also demonstrate considerable potential for the problem of illicit transaction detection in cryptocurrency networks. One of their key advantages is the ability to capture long-range dependencies through global self-attention mech-

anisms. Unlike traditional GCNs, which are limited to aggregating information from a node's immediate neighbors, Graph Transformers can model interactions between distant nodes—an essential capability when tracing complex laundering schemes that span multiple hops across the transaction graph.

Furthermore, Graph Transformers can be adapted to handle temporal information by incorporating positional or time-aware embeddings, making them particularly useful in identifying evolving patterns such as peeling chains or layering tactics often used in money laundering. Their high expressive power allows them to learn complex, non-linear relationships within the data, which can be beneficial in uncovering subtle indicators of fraud that might be missed by simpler models.

Another important strength lies in their adaptive attention mechanism, which enables the model to focus more effectively on suspicious or high-impact nodes, even if these are not highly central within the network. This dynamic focus offers flexibility and robustness, especially in rapidly changing or large-scale blockchain environments. Additionally, Graph Transformers support the integration of rich feature sets—such as transaction metadata, timestamps, and address properties—making them versatile tools for modeling multi-dimensional blockchain data.

Although they require more computational resources and careful tuning compared to GCNs, Graph Transformers remain a powerful and promising direction for future work, particularly when combined with domain-specific enhancements.

5.3 Graph Transformer

The Graph Transformer applies a transformer-inspired convolution operation to graph-structured data. Its architecture consists of multiple stacked **Transformer Convolution layers**, interleaved with nonlinear activation functions and dropout for

regularization.

5.3.1 Working Mechanism

- The model begins with an **Input Graph** consisting of nodes and edges, where each node contains initial feature vectors.
- The graph is passed through successive **Transformer Convolution** layers.
- Each convolutional layer focuses on neighboring nodes to compute updated embeddings.
- The output is passed through a **Log Softmax** layer to predict class probabilities for each node.

5.3.2 Feature Transformation

Transformer convolution layers modify node features by attending over the neighborhood. They learn which neighbors are most important for a node through a multi-head attention mechanism. This leads to:

- Better representation of complex dependencies.
- Emphasis on task-relevant neighbors.
- Dynamic aggregation rather than fixed (e.g., mean or sum).

5.3.3 Insights and Advantages

- Captures long-range dependencies.
- Learns contextual embeddings better than traditional GCN.
- Provides interpretability through attention weights.

5.3.4 Pipeline Diagram

Here is the pipeline diagram illustrating the various stages of the model:

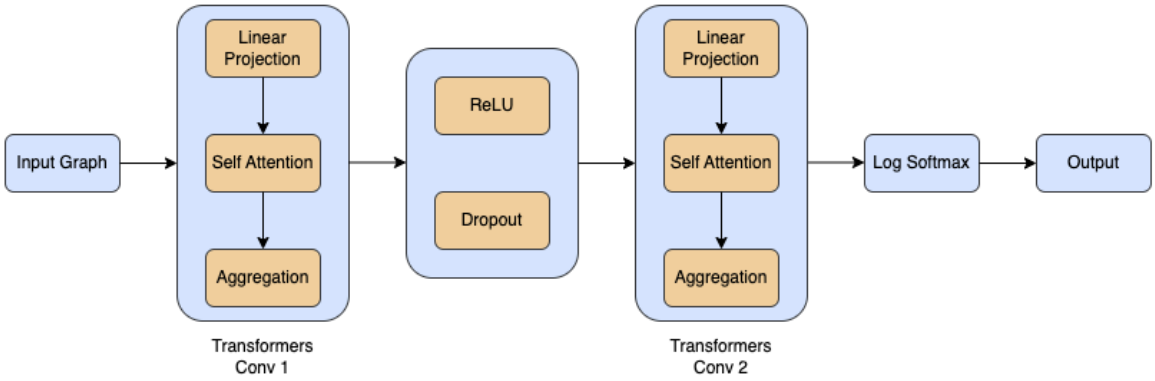


Figure 5.1: Feature selection pipeline using Random Forest and `scikit-learn` before input to GCN.

The diagram above illustrates the pipeline architecture of a Graph Transformer model, breaking down the overall data flow and the internal workings of its key components.

The top row of the diagram shows the end-to-end flow of data through the model:

- **Input Graph**: The raw graph data enters the model.
- **Transformer Conv 1**: The first graph transformer convolutional layer processes the input.
- **ReLU + Dropout**: Nonlinear activation (ReLU) is applied, followed by dropout for regularization.
- **Transformer Conv 2**: A second graph transformer convolutional layer further processes the data.
- **Log Softmax**: The output is passed through a log softmax function, commonly used for classification tasks.

- **Output Data**: The final output of the model, such as class probabilities or predictions.

Each blue box represents a major stage in the pipeline, and arrows indicate the direction of data flow from one stage to the next.

The bottom part of the diagram zooms into the internal processing steps of each Transformer Conv layer (Conv 1 and Conv 2):

- **Linear Projection**: The input features are projected into a higher-dimensional space, preparing them for attention calculations.

- **Self Attention**: The model computes attention scores, allowing each node in the graph to gather information from other nodes based on learned relationships.

- **Aggregation**: The attended features are aggregated, combining information from different nodes to produce an updated representation.

This sequence occurs in both Transformer Conv 1 and Transformer Conv 2, as shown in the diagram.

This structure enables the model to learn complex, context-aware representations for graph data, leveraging the power of self-attention at each convolutional stage.

5.4 Graph Transformer with GPE

5.4.1 Motivation

Despite the advantages of the base Graph Transformer, it lacks explicit encoding of positional information, which is crucial in graphs where relative positioning matters. To address this, we introduce the Graph Positional Encoding (GPE).

5.4.2 What is GPE?

Graph Positional Encoding is derived from the **eigenvectors of the graph Laplacian**. It encodes:

- The structural identity of each node.
- The relative position of nodes in the global graph structure.
- Latent relationships beyond immediate neighborhoods.

5.4.3 Architecture Enhancements

The enhanced model includes a parallel branch for GPE:

- The GPE vector is combined with the original node features.
- A **Linear Projection** maps the input to a lower-dimensional space.
- **Self-Attention** and **Aggregation** layers compute a position-aware embedding.
- The result is merged with the output of the Transformer Convolution path.

5.4.4 Feature Impact and Advantages

- Enhances node features with structural context.
- Provides better awareness of a node's position in the global graph.
- Can distinguish structurally similar but distant nodes.
- Leads to marginal improvements in classification F1-score.

Table 5.1: Comparison of F1-Scores across different models

Model	F1-Score
GCN	74.18
GAT	76.27
GraphSage	78.38
Transformer	76.09
Transformer with GPE	76.20

5.5 Comparative Results

The results show that **GraphSage** achieves the highest F1-Score (78.38), outperforming both the standard Transformer (76.09) and Transformer with GPE (76.20). Among the transformer-based models, the addition of GPE slightly improves over the vanilla Transformer. GAT and Transformer models perform comparably, while GCN yields the lowest F1-score among the evaluated methods. These results highlight the effectiveness of GraphSage for the given task and suggest that while transformer-based approaches are competitive, further enhancements may be required to surpass the performance of GraphSage.

Chapter 6

Graph Augmentation using Centrality

While Graph Transformer models demonstrated strong potential in capturing long-range dependencies and temporal patterns within the transaction graph, their performance in the context of illicit transaction detection was not consistently superior. One of the key limitations observed was the model's general-purpose attention mechanism, which, despite being expressive, does not inherently prioritize structurally significant nodes involved in fraudulent activities.

This observation motivated the exploration of a more domain-informed approach—specifically, graph augmentation using classical centrality measures such as betweenness and eigen

6.1 Betweenness Centrality

Betweenness centrality measures how often a node appears on the shortest paths between other nodes in a network. In simple terms, a node with high betweenness centrality acts as a bridge connecting different parts of the network. Removing such

a node could significantly disrupt communication or flow within the network.

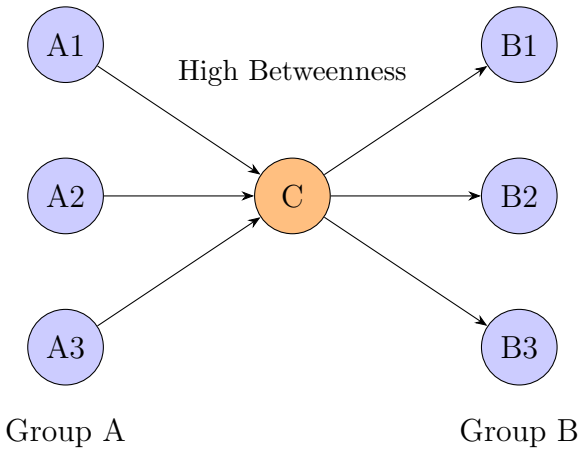


Figure 6.1: Node C lies on the shortest paths between nodes in Group A and Group B, indicating high betweenness centrality.

In the diagram above, the node in the center plays a critical role in linking two separate groups. This reflects a real-world scenario, such as a hub airport connecting two continents or a key server routing data in a network.

6.1.1 An Introduction

Betweenness centrality is a key metric in network analysis that measures how much a node lies on the shortest paths between other nodes in a graph. Formally, the betweenness centrality $C_B(v)$ of a node v is defined as:

$$C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

pwhere:

- σ_{st} is the total number of shortest paths from node s to node t,
- $\sigma_{st}(v)$ is the number of those paths that pass through node v.

In essence, this metric quantifies the importance of a node in facilitating communication between other nodes in the network.

6.1.2 Betweenness Centrality in Bitcoin Networks

In the context of the Bitcoin transaction network, nodes can represent **individual transactions**, and edges indicate the **flow of Bitcoin** from one transaction to another. When analyzing illicit behavior, the network is often structured such that **nodes represent illicit transactions**, and directed edges capture how funds are transferred across these suspicious transactions.

Betweenness centrality in such a network quantifies how often a node (transaction) appears on the shortest paths between other pairs of nodes. A transaction with high betweenness centrality acts as a *key Intermediary*, facilitating the flow of funds and potentially bridging otherwise disconnected parts of the network.

These central transactions are particularly interesting in financial forensics and anti-money laundering (AML) investigations. They may:

- Serve as **mixing points** or **funnels** where multiple streams of illicit funds converge.
- Be used to **split or merge** amounts for obfuscating the origin or destination of funds.
- Act as **stepping stones** in chains of transactions, helping obscure traceability.

Identifying such transactions can reveal hidden coordination strategies malicious actors use to avoid detection. Moreover, targeting high-betweenness nodes may help disrupt the illicit flow of funds by:

- Increasing the length or complexity of alternative transaction paths.

- Making the laundering process more costly or easier to trace.

In summary, betweenness centrality offers a powerful graph-based perspective to uncover **critical links** in the flow of suspicious Bitcoin activity, supporting more effective monitoring and intervention efforts.

6.1.3 Illicit Activity and Centrality Metrics

Betweenness centrality is particularly relevant in the analysis of illicit financial activity for the following reasons:

1. **Intermediary Roles:** Illicit wallets often attempt to mask the origin or destination of funds by routing them through intermediary wallets. A wallet with high betweenness centrality may be a crucial link in these obfuscation chains.

2. **Detection of Money Laundering:** During the layering phase of money laundering, transactions are often split into smaller amounts and routed through multiple wallets. Wallets with high betweenness centrality can indicate hubs that consolidate or disperse funds, usually associated with mixers or laundering services.

3. **Structural Importance:** Even if a wallet does not initiate or receive large funds, its high betweenness centrality can reveal its structural importance in facilitating network-wide transactions, making it a key target for investigative efforts.

4. **Early Warning Indicators:** A wallet that rapidly gains high betweenness centrality may be part of a newly activated illicit operation. Monitoring such nodes can provide early alerts for suspicious activity.

6.1.4 Example

Consider a transaction path from wallet A (origin of illicit funds) to wallet F (a cash-out point at an exchange). If wallet C frequently lies on the shortest transaction paths from A to F, as well as other similar paths in the network, it suggests that:

- Wallet C is strategically positioned,
- It may act as a mixer or bridge in a laundering network,
- It plays a non-trivial role in enabling flow between isolated clusters.

6.1.5 How to Compute Betweenness Centrality

Here is a step-by-step approach:

1. Initialize centrality score $C_B(v) = 0$ for all nodes v.
2. For each node s in the graph:
   (a) Compute shortest paths from s to all other nodes.
   (b) Count the number of shortest paths σ_{st} between each node pair (s, t).
   (c) For each intermediate node v, count how many shortest paths pass through v, denoted $\sigma_{st}(v)$.
   (d) Update centrality:
   $$C_B(v) += \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$
3. Normalize the scores:
   $$C_B(v) = \frac{C_B(v)}{(N-1)(N-2)}$$
   where N is the number of nodes in the graph.

6.1.6 Pipeline Diagram

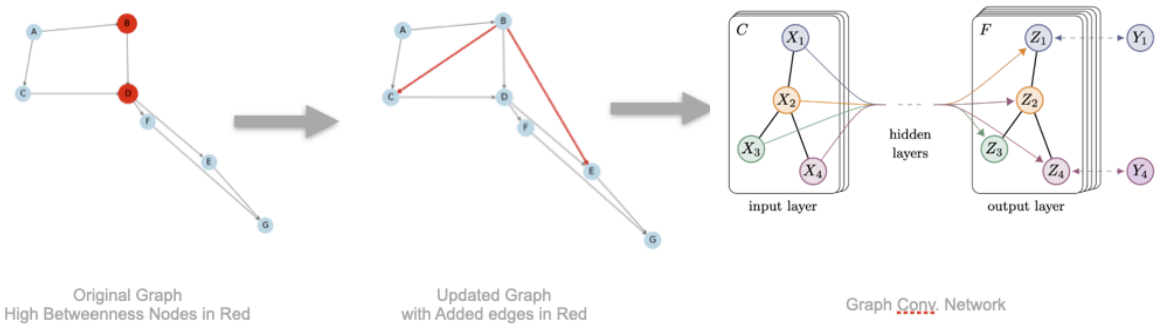


Figure 6.2: Edge Augmentation Based on Betweenness Centrality and Its Integration with a GCN

Figure 6.2 presents the complete pipeline that enhances the graph structure using betweenness centrality before passing it into a Graph Convolutional Network (GCN) for classification.

Original Graph (Left): The initial graph consists of nodes (representing transactions) and edges (representing Bitcoin flow). Nodes highlighted in red have high betweenness centrality, indicating they frequently lie on the shortest paths between other nodes and act as crucial intermediaries.

Updated Graph (Middle): Based on the computed betweenness centrality, new edges (in red) are added to the graph. These edges aim to capture latent relationships and improve information propagation in the network. This graph augmentation step enhances the graph's expressiveness and structural richness.

Graph Convolutional Network (Right): The augmented graph is then passed into a GCN. The input layer receives node features (e.g., transaction statistics), and the GCN aggregates neighborhood information through multiple hidden layers. The output layer provides class predictions, such as whether a transaction is illicitfeatures.

This pipeline demonstrates how incorporating structural insights, such as Central-

ity, can lead to better learning and more accurate detection of suspicious activity in the Bitcoin transaction network.

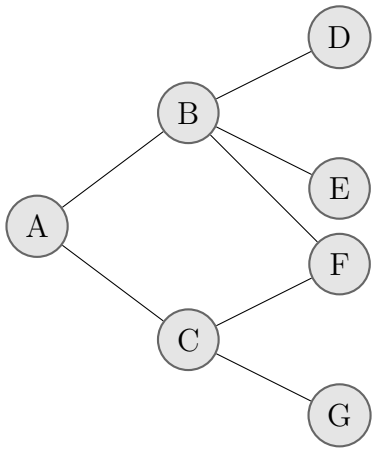
6.2 Eignvector Centrality

Eigenvector centrality is a way to measure how **important** a node is in a network. Unlike simple centrality measures that only count how many connections a node has, eigenvector centrality says:

> "A node is important if connected to other important nodes."

In social terms, you are popular not just because you have many friends but also because your friends are popular.

Example:



In this graph:

- Node A has only two connections, but B and C are connected to many other nodes.
- So A's importance increases because it's connected to B, C, and many others.

- Node B has high eigenvector centrality because it's connected to other nodes like D, E, and F, which help boost its importance.

Eigenvector centrality captures this recursive idea of importance.

6.2.1 An Introduction

In network analysis, centrality metrics are widely used to identify the graph's most influential or essential nodes. Among these, **Eigenvector Centrality** offers a more nuanced perspective compared to basic metrics such as degree centrality. While degree centrality counts how many direct connections a node has, eigenvector centrality goes a step further by considering the quality of those connections. Specifically, it assigns relative scores to all nodes in the network based on the principle that connections to highly connected nodes contribute more to a node's score than equal connections to less connected nodes.

The concept is based on the idea of *recursive influence*: a node is essential if connected to other important nodes. This makes eigenvector centrality particularly powerful in identifying nodes that are not just locally well-connected but are also situated in influential regions of the network structure. It has been widely applied in diverse domains, including social network analysis, citation networks, web page ranking (as in Google's PageRank algorithm), and more recently, in financial transaction networks such as Bitcoin, to identify nodes that may play central roles in the flow of value or illicit activity.

Eigenvector centrality is computed using the leading eigenvector of the network's adjacency matrix, providing a mathematically grounded and scalable method for evaluating node influence in complex networks.

6.2.2 Eigenvector Centrality in Bitcoin Networks

As a decentralized digital currency, Bitcoin relies on a peer-to-peer network where transactions are recorded on a public ledger known as the blockchain. Each transaction involves at least one sender and receiver, forming a complex network of address interactions over time. This transactional structure can be naturally modeled as a directed graph, where nodes represent wallet addresses and edges represent bitcoin flow from one address to another.

Applying **Eigenvector Centrality** to such a network allows us to identify the most influential addresses based on their position and connections within the network. Unlike simple metrics such as transaction count or volume, eigenvector centrality considers direct and indirect interactions. That is, an address gains importance not only by sending or receiving many transactions but also by being connected to other addresses that are central to the network.

In the context of Bitcoin, addresses with high eigenvector centrality may represent major exchanges, popular service providers, or potentially illicit entities acting as central hubs in laundering schemes. Thus, eigenvector centrality is a valuable analytical tool for uncovering hidden structures and influence patterns in the Bitcoin transaction graph.

By focusing on these structurally important nodes, researchers and analysts can better understand the flow of value within the Bitcoin ecosystem and potentially detect suspicious or illegal activities.

6.2.3 Relevance of Eigenvector Centrality in Illicit Financial Activity Analysis

Eigenvector centrality is a powerful tool in network analysis, particularly suitable for uncovering influential nodes in complex transaction networks such as the Bitcoin blockchain. Its relevance in detecting illicit financial activity arises from several key characteristics:

1. **Influence Beyond Direct Connections:** Eigenvector centrality accounts not only for the number of connections a node has but also the quality of those connections. In illicit finance, key participants often use intermediaries to mask their activity. This measure highlights nodes connected to other influential nodes, making detecting those acting as hidden hubs possible.

2. **Detection of Coordinated Activity:** Criminal networks often involve multiple cooperating addresses that may not individually stand out. Eigenvector centrality helps reveal densely connected subgraphs that interact significantly with the core network, suggesting potential coordinated behavior.

3. **Identifying Money Laundering Funnels:** High-centrality nodes often serve as choke points through which substantial value flows. These nodes can be critical steps in the layering phase of money laundering. Analyzing them provides insights into the transaction paths and illicit funds' potential origin and destination.

4. **Prioritization for Investigation:** With a large number of addresses on the Bitcoin network, it becomes essential to prioritize nodes for detailed scrutiny. Eigenvector centrality can be used as a ranking mechanism to highlight nodes with systemic importance in the transaction network.

5. **Resilience to Evasion Tactics:** Criminals may distribute transactions across numerous addresses to evade detection. Eigenvector centrality, by measuring the structural influence of a node rather than surface-level activity, is robust against such evasion strategies and can still identify underlying influential actors.

6.2.4 Example

Consider a segment of the transaction network where wallet A is connected to high-traffic wallets B and C, while wallet D is only connected to wallet A. Although wallet D directly connects to A, its overall importance is limited because A is not among the most central wallets.

In contrast, wallets B and C are each connected to several other highly active wallets in the network. In this scenario, eigenvector centrality highlights that:

- Wallets B and C have high influence due to their proximity to other important wallets,
- Wallet A inherits some centrality from its links to B and C,
- Wallet D, despite being connected, ranks low in influence because its only connection is to a moderately central wallet.

This illustrates that eigenvector centrality rewards wallets not just for their number of connections but for how central their neighbors are, helping to identify influential nodes within the broader laundering network.

6.2.5 How to Compute Eigenvector Centrality

Given a Bitcoin transaction graph $G = (V, E)$ with n nodes and edges, this algorithm computes the eigenvector centrality score for every node in the graph.

1. **Input:**

   - Adjacency matrix $A \in \mathbb{R}^{n \times n}$ representing the connections between nodes,
   - Convergence threshold $\epsilon > 0$ (e.g., 10^{-6}),
   - Maximum number of iterations T.

2. **Initialize:**

   - Set the initial centrality vector $x^{(0)} = [1, 1, \ldots, 1]^T \in \mathbb{R}^n$,
   - Set iteration counter $k = 0$.

3. **Iterative Computation:**

   (a) Repeat until convergence or until $k = T$:

   - Compute the new centrality estimate: $x^{(k+1)} = Ax^{(k)}$,
   - Normalize the vector: $x^{(k+1)} = \dfrac{x^{(k+1)}}{\|x^{(k+1)}\|_2}$,
   - If $\|x^{(k+1)} - x^{(k)}\|_2 < \epsilon$, stop the iteration,
   - Else, increment $k = k + 1$.

4. **Output:**

   - The final vector $x^{(k+1)} = [c_1, c_2, \ldots, c_n]^T$ where each c_i is the eigenvector centrality score of node $v_i \in V$.

6.2.6 Pipeline Diagram

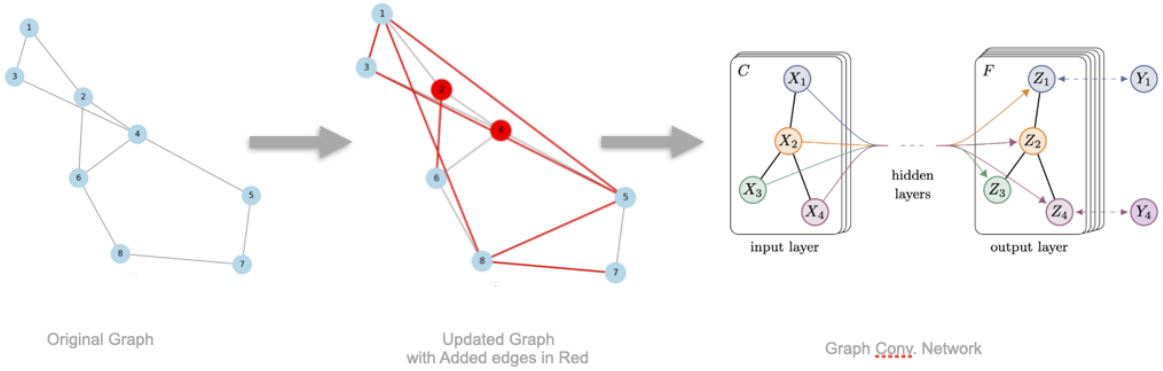


Figure 6.3: Pipeline of Eigenvector Centrality-Based Graph Augmentation and Classification

The pipeline illustrated in Figure 6.3 represents the overall workflow of our Eigenvector Centrality-based approach to identifying illicit nodes in the Bitcoin transaction network.

The process begins with an **original transaction graph** where nodes represent wallet addresses and edges represent transactions between them. Next, we calculate each node's **Eigenvector Centrality** to determine its importance in the network. Based on a threshold, we identify highly central nodes and strategically **add new edges** (highlighted in red) to enhance the graph's connectivity. This results in an **updated graph** with improved structural representation of influential regions.

Finally, the augmented graph is fed into a **Graph Convolutional Network (GCN)** for node classification. The GCN leverages the topological structure and node features to predict whether a node (wallet) is associated with illicit or licit activity. This pipeline boosts the model's capability to capture relational patterns tied to criminal behavior in the network.

6.3 Experiments

In this section, we evaluate the effectiveness of our proposed graph augmentation technique using eigenvector centrality in the context of a node classification task. The objective is to determine whether the newly generated graph structures, enhanced by centrality-based edge additions, lead to improved performance when utilized in a Graph Convolutional Network (GCN) model.

We compare the results of our proposed method with baseline approaches introduced in the FraudLens framework [11], which include Edge Affinity (EA) and Edge based on Node Features (ENF) graphs, both trained using the GraphSAGE architecture. In the baseline setting, edges are randomly dropped during training with a fixed probability p (dropout-based regularization), which simulates a perturbation-based augmentation approach.

6.3.1 Parameter Settings

To construct the augmented graphs based on Centrality, we apply the following configuration:

- **Centrality Tolerance:** When generating edges based on betweenness and eigenvector centrality, we define a relative tolerance threshold of 10^{-7}. This tolerance controls the sensitivity when comparing centrality values of node pairs. If the absolute difference in Centrality between two nodes is within this threshold, we consider them similar in influence and thus connect them with a new edge.

- **GCN Configuration:** We implement a 2-layer Graph Convolutional Network (GCN) using the GCNConv operator. Each layer is followed by a ReLU activation function to introduce non-linearity. The network is optimized using

standard cross-entropy loss on the node classification task.

- **GraphSAGE Configuration:** For comparison, we follow the settings used in [11] for the GraphSAGE model. The architecture consists of 5 SAGEConv layers with a hidden feature size 32. Each layer uses ReLU activation, and training is regularized using hyperparameters $\alpha_g = 0.0001$, $\alpha_f = 0.01$, $\beta_g = 1.0$, $s = 5$, and $T = 1$.

6.3.2 Training Setup

We train each model for 1000 epochs using the Adam optimizer with early stopping based on validation loss. The dataset is temporally partitioned based on transaction timestamps to simulate a realistic evaluation scenario:

- **Training Window:** Timesteps 7 to 11
- **Validation Window:** Timesteps 12 to 17
- **Testing Window:** Timesteps 18 to 49

These splits ensure that the models are evaluated temporally consistently and prevent data leakage across time.

6.3.3 Evaluation Metric

The primary metric used to assess model performance is the F1-score, calculated separately for multiple testing windows for evaluating robustness over time. We report per-window F1-scores to capture fluctuations in performance.

6.4 Results and Observations

In this section, we comprehensively evaluate the proposed centrality-based graph augmentation methods for detecting illicit transactions in the Elliptic++ dataset. We focus on analyzing how these graph construction techniques influence the performance of node classification models across multiple time windows, particularly in terms of the F1-score. We compare our approaches with existing baselines to highlight improvements and trade-offs.

6.4.1 Comparison with Baseline Graph Constructions

Table 6.1 summarizes the classification performance across multiple time intervals using different graph construction strategies and GNN architectures. The baseline models, G_{ENF} and G_{EA}, are derived from the FraudLens framework [11] and use the GraphSAGE architecture with random edge dropout (probability $p = 0.2$). These are compared with our proposed graph variants, $G_{\text{betweenness}}$ and $G_{\text{eigenvector}}$, both trained using a GCN model.

The results show that the centrality-based graphs exhibit strong and consistent performance, especially in early to mid-range time intervals:

- $G_{\textbf{betweenness}}$ **(GCN)** achieves the highest F1-score (79.11) in the 23–26 time step and maintains competitive scores across other intervals.
- $G_{\textbf{eigenvector}}$ **(GCN)** yields the best F1-score (83.21) in the 18–22 window and shows robust performance across later intervals such as 78.66 (31–34), 77.05 (35–39), and 74.83 (40–43).
- Both proposed methods outperform G_{EA} across all time intervals by a significant margin, indicating that random edge augmentation (as used in EA) is less

effective at capturing meaningful graph structure.

- While G_{ENF} shows strong performance in the 27–30 window (83.35), it suffers from greater fluctuations across time steps compared to centrality-based methods.

These findings demonstrate that incorporating global structural properties through centrality metrics allows the model to capture persistent topological roles (e.g., addresses acting as mixers, collectors, or intermediaries), crucial in tracking illicit behavior patterns. In dynamic financial environments, such as Bitcoin networks, this structural resilience offers better generalization, especially when fraud patterns evolve due to external shocks like darknet shutdowns or changes in laundering techniques.

Table 6.1: F1-scores with random edges dropped ($p = 0.2$) for GraphSAGE and comparison with our centrality-based results using GCN

Graph	GNN-Arch/Model	18–22	23–26	27–30	31–34	35–39	40–43
G_{ENF}	GraphSAGE ($p = 0.2$)	80.27	75.07	**83.35**	74.44	76.96	**77.23**
G_{EA}	GraphSAGE ($p = 0.2$)	59.18	60.84	67.65	59.07	67.13	68.02
$G_{\text{betweenness}}$	GCN	83.20	**79.11**	78.98	78.50	74.60	68.40
$G_{\text{eigenvector}}$	GCN	**83.21**	78.97	79.47	**78.66**	**77.05**	74.83

6.4.2 Effect of Tolerance on Graph Generation and Model Accuracy

Table 6.2 investigates the sensitivity of our methods to the tolerance parameter, which governs the threshold used for creating edges between nodes with similar centrality values. Specifically, we examine how different tolerances (10^{-3}, 10^{-5}, 10^{-7}, and 10^{-10}) influence the quality of the generated graph and the downstream classification accuracy.

Table 6.2: F1-scores of Betweenness and Eigenvector centrality-based graphs for different tolerance values using GCN architecture

Graph	Tolerance	18–22	23–26	27–30	31–34	35–39	40–43
$G_{\text{betweenness}}$	10^{-3}	81.74	77.11	77.14	69.51	70.71	69.91
$G_{\text{betweenness}}$	10^{-5}	82.17	**79.95**	78.11	67.91	69.17	62.22
$G_{\text{betweenness}}$	10^{-7}	**83.20**	79.11	**78.98**	**78.50**	**74.60**	68.40
$G_{\text{betweenness}}$	10^{-10}	82.38	77.75	78.59	71.68	73.10	**71.03**
$G_{\text{eigenvector}}$	10^{-3}	81.04	76.56	75.84	65.92	64.95	58.52
$G_{\text{eigenvector}}$	10^{-5}	81.65	**80.43**	**80.27**	**79.51**	**77.57**	73.93
$G_{\text{eigenvector}}$	10^{-7}	79.02	77.49	77.29	76.40	76.33	73.89
$G_{\text{eigenvector}}$	10^{-10}	**83.21**	78.97	79.47	78.66	77.05	**74.83**

- For graphs constructed using betweenness centrality ($G_{\text{betweenness}}$), the optimal performance is achieved with a tolerance of 10^{-7}. This setting produces the highest F1-score of 83.20 in the 18–22 interval and also yields competitive performance in intervals 27–30 (78.98) and 31–34 (78.50). Lower tolerances such as 10^{-10}, slightly reduce performance in some intervals and may incur additional computational cost without proportionate benefit. Very loose tolerances (10^{-3}) underperform in later intervals, likely due to over-connection and noise in the graph.

- For $G_{\text{eigenvector}}$, the best F1-score of 83.21 also occurs at 10^{-10}, but overall consistency across all time steps is better at 10^{-5}. With 10^{-5}, we observe high scores such as 80.43 (23–26), 80.27 (27–30), and 79.51 (31–34), indicating strong generalization and stability. In contrast, higher tolerances (e.g., 10^{-3}) result in lower

and inconsistent performance, which could be due to imprecise edge formation connecting dissimilar nodes.

Summary of Observations

- Centrality-based edge augmentation improves classification accuracy and robustness compared to baseline methods like EA and ENF.
- Betweenness centrality with tolerance 10^{-7} provides the best trade-off between accuracy and stability.
- Eigenvector centrality benefits from a slightly lower tolerance (10^{-10}) in peak intervals but shows more consistent performance with 10^{-5}.
- Proper tuning of the tolerance parameter is critical to ensuring optimal graph structure without excessive noise or sparsity.
- Centrality-based approaches are better suited to real-world fraud detection, where structural roles in the network (e.g., intermediaries, mixers) are more persistent than specific transaction behaviors.

These observations strongly support our hypothesis that incorporating global structural information through centrality measures leads to superior and more reliable node classification outcomes in the context of illicit activity detection in cryptocurrency networks.

Chapter 7

Conclusion and Scope for Future Work

7.1 Conclusion

In this thesis, we explored the use of graph-based machine learning models for detecting illicit transactions in Bitcoin networks. By leveraging the structural properties of transaction graphs and augmenting them using centrality measures such as betweenness and eigenvector centrality, we have demonstrated that incorporating graph-based features significantly enhances the performance of Graph Convolutional Networks (GCNs) in detecting illicit activities.

Our experimental results show that centrality-based graph construction strategies, particularly those based on betweenness and eigenvector centrality, consistently outperform other graph construction methods like Edges Based on Affinity (EA) and Edges Based on Node Features (ENF). These centrality measures enable the model to capture crucial network relationships that may otherwise go undetected, especially when identifying transactions involving intermediary wallets or clusters of illicit activity. Additionally, we observed that selecting an appropriate tolerance parameter for

centrality calculations is critical to optimizing model performance.

We also investigated two other complementary approaches: (1) a feature selection strategy using Random Forest followed by a CNN classifier, and (2) a Graph Transformer-based model capable of capturing long-range dependencies and temporal patterns in the transaction graph. While the Random Forest + CNN pipeline showed performance improvements when used with simpler GCN models, it did not scale as effectively with more complex GNN architectures. Similarly, the Graph Transformer approach demonstrated competitive performance and was effective in modeling temporal dependencies; however, it lacked the structural sensitivity and consistent accuracy provided by our centrality-based augmentation strategy.

Among the three approaches studied, the centrality-based graph augmentation approach emerged as the most robust and generalizable. It consistently yielded higher F1-scores across multiple time steps and outperformed both the Random Forest + CNN and Graph Transformer approaches in detecting complex patterns of illicit activity. This validates our hypothesis that structural augmentation using centrality metrics provides critical insights into transaction behaviors that are essential for accurate classification.

In summary, integrating centrality measures into graph-based models—particularly in conjunction with GCNs—provides a powerful and scalable method for identifying illicit transactions in Bitcoin networks. This work contributes to the growing field of blockchain analytics by offering a novel, interpretable, and effective framework that combines classical network science with modern machine learning techniques for fraud detection.

7.2 Scope for Future Work

While the results presented in this thesis show promising outcomes, several areas could be explored further to enhance the approach and extend its applicability.

7.2.1 Integration of Additional Graph Augmentation Techniques

Future work could explore the integration of multiple graph augmentation techniques, such as PageRank, Katz centrality, or other higher-order centrality measures, to enhance the ability of the model to capture more comprehensive relational information. By incorporating these additional graph augmentations, the model could potentially identify more subtle and complex patterns of illicit behavior that are currently missed by betweenness and eigenvector centrality alone.

7.2.2 Generalization Across Different Blockchain Networks

The current study focuses on Bitcoin transactions, but the proposed methodology can be extended to other blockchain networks such as Ethereum, Litecoin, and others. Each blockchain may have different transaction structures and types of illicit behavior, and testing the model's generalizability on diverse datasets can help refine and validate the approach across multiple platforms.

7.2.3 Real-Time Fraud Detection

One of the major challenges in blockchain-based fraud detection is real-time monitoring of transactions. Future research could investigate the feasibility of adapting the proposed method for real-time detection of illicit transactions. This would require optimizing the computational complexity of graph construction and machine learn-

ing models to handle high throughput of transactions. Leveraging parallel computing techniques, such as GPU-based graph processing libraries or distributing the workload across multiple nodes, could help address this challenge.

7.2.4 Interpretability of Graph-Based Models

As the complexity of machine learning models increases, particularly in deep learning architectures like GCNs, interpretability becomes a critical factor. Future work could focus on developing techniques to make the results of GCN models more interpretable to users and stakeholders in fraud detection systems. For instance, attention mechanisms or feature importance analysis could help explain which nodes or edges are most critical for detecting illicit transactions, improving trust in the model's decisions.

7.2.5 Integration with Other Types of Data

Currently, the model focuses on transaction data and network structure. However, combining this approach with other data types, such as off-chain metadata (e.g., exchange records, wallet behavior patterns, or social media activity), could further enhance the detection capabilities. By integrating multi-modal data sources, the model could gain a deeper understanding of illicit activity beyond just the transaction graph.

7.2.6 Further Evaluation on Large-Scale Datasets

Lastly, while we evaluated our model on a Bitcoin transaction dataset, further work could involve testing the scalability and robustness of the approach on even larger datasets. As blockchain networks grow, the size of transaction graphs increases, making it essential to evaluate the scalability of the proposed methods. This includes testing the model on high-dimensional graphs with millions of nodes and edges to

ensure the strategies are computationally feasible.

7.3 Final Remarks

The detection of illicit transactions in blockchain networks is a crucial aspect of ensuring the integrity and trustworthiness of these decentralized systems. In this work, we have shown that leveraging graph-based features, particularly centrality measures, in conjunction with machine learning models such as GCNs, can lead to significant improvements in detecting fraudulent transactions. The promising results and the potential for future advancements open new avenues for research in blockchain analytics and fraud detection, making it an exciting area for further exploration.

Bibliography

[1] Alarab, I., and Prakoonwit, S. Effect of data resampling on feature importance in imbalanced blockchain data: Comparison studies of resampling techniques. *Data Science and Management 5*, 2 (2022), 66–76.

[2] Alarab, I., Prakoonwit, S., and Nacer, M. I. Comparative analysis using supervised learning methods for anti-money laundering in bitcoin. In *Proceedings of the 2020 5th international conference on machine learning technologies* (2020), pp. 11–17.

[3] Alarab, I., Prakoonwit, S., and Nacer, M. I. Competence of graph convolutional networks for anti-money laundering in bitcoin blockchain. In *Proceedings of the 2020 5th international conference on machine learning technologies* (2020), pp. 23–27.

[4] Biswas, S., Wardat, M., and Rajan, H. The art and practice of data science pipelines: A comprehensive study of data science pipelines in theory, in-the-small, and in-the-large. In *Proceedings of the 44th International Conference on Software Engineering* (2022), pp. 2091–2103.

[5] Chen, J., Pareja, A., Domeniconi, G., Ma, T., Suzumura, T., Kaler, T., Schardl, T. B., and Leiserson, C. E. Evolving graph convolutional networks for dynamic graphs, Dec. 27 2022. US Patent 11,537,852.

[6] Dwivedi, V. P., and Bresson, X. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2003.00982* (2020).

[7] Elmougy, Y., and Liu, L. Demystifying fraudulent transactions and illicit nodes in the bitcoin network for financial forensics. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (2023), pp. 3979–3990.

[8] Graves, A., and Graves, A. Long short-term memory. *Supervised sequence labelling with recurrent neural networks* (2012), 37–45.

[9] Kursa, M. B., and Rudnicki, W. R. The all relevant feature selection using random forest. *arXiv preprint arXiv:1106.5112* (2011).

[10] Lo, W. W., Kulatilleke, G. K., Sarhan, M., Layeghy, S., and Portmann, M. Inspection-l: self-supervised gnn node embeddings for money laundering detection in bitcoin. *Applied Intelligence 53*, 16 (2023), 19406–19417.

[11] Nicholls, J., Kuppa, A., and Le-Khac, N.-A. Fraudlens: Graph structural learning for bitcoin illicit activity identification. In *Proceedings of the 39th Annual Computer Security Applications Conference* (2023), pp. 324–336.

[12] Park, W., Chang, W., Lee, D., Kim, J., and Hwang, S.-w. Grpe: Relative positional encoding for graph transformer. *arXiv preprint arXiv:2201.12787* (2022).

[13] Singh, A., Gupta, A., Wadhwa, H., Asthana, S., and Arora, A. Temporal debiasing using adversarial loss based gnn architecture for crypto fraud detection. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)* (2021), IEEE, pp. 391–396.

[14] Song, J., Zhang, S., Zhang, P., Park, J., Gu, Y., and Yu, G. Illicit social accounts? anti-money laundering for transactional blockchains. *IEEE Transactions on Information Forensics and Security* (2024).

[15] Vassallo, D., Vella, V., and Ellul, J. Application of gradient boosting algorithms for anti-money laundering in cryptocurrencies. *SN Computer Science 2*, 3 (2021), 143.

[16] Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[17] Wang, S., Aggarwal, C., and Liu, H. Using a random forest to inspire a neural network and improving on it. In *Proceedings of the 2017 SIAM international conference on data mining* (2017), SIAM, pp. 1–9.

[18] Weber, M., Domeniconi, G., Chen, J., Weidele, D. K. I., Bellei, C., Robinson, T., and Leiserson, C. E. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591* (2019).

[19] Weber, M., Domeniconi, G., Chen, J., Weidele, D. K. I., Bellei, C., Robinson, T., and Leiserson, C. E. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591* (2019).

[20] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems 32*, 1 (2020), 4–24.

[21] Yun, S., Jeong, M., Kim, R., Kang, J., and Kim, H. J. Graph transformer networks. *Advances in neural information processing systems 32* (2019).

[22] Zhang, S., Tong, H., Xu, J., and Maciejewski, R. Graph convolutional networks: a comprehensive review. *Computational Social Networks 6*, 1 (2019), 1–23.