

VIETNAM NATIONAL UNIVERSITY,
HANOI INTERNATIONAL SCHOOL



Topic: Anti-Money Laundering Detection with Graph Neural Networks (GNN)

| | |
|---------------|---------------------------|
| Class | : INS306302 |
| Lecturer | : TS Trương Công Đoàn |
| Subject | : Advance Data Analytics |
| Student name | : Phí Hải Việt – 20071000 |
| Student Class | : BDA2020B |

Hanoi, 2024

Table of Contents

Table of Contents

| | |
|--|----|
| I. Abstract..... | 3 |
| II. Introduction | 4 |
| 1. The Anti-Money Laundering (AML) Challenge | 4 |
| 2. Defining Fraud in Money Laundering..... | 4 |
| 3. Synthetic Datasets..... | 5 |
| 4. Why Apply Graph Neural Networks (GNNs) for AML Detection? | 6 |
| 5. Research Focus and Methodology | 7 |
| III. Survey | 7 |
| IV. Data | 9 |
| 1. Dataset Overview..... | 9 |
| 2. Data Structure | 12 |
| V. Methods | 12 |
| 1. Why Apply Graph Neural Networks (GNNs)? | 12 |
| 2. Graph Construction and Feature Engineering | 13 |
| 3. Overview of Graph Attention Networks (GAT) | 14 |
| 4. Application of GAT to the IBM Dataset..... | 15 |
| 5. Comparison with GCN and GIN..... | 16 |
| 6. Evaluation Metrics | 17 |
| VI. Implementation | 18 |
| 1. Environment Setup..... | 18 |
| 2. Data Loading and Preprocessing..... | 18 |
| 3. Graph Construction..... | 19 |
| 4. GAT Model Architecture..... | 20 |
| 5. Training the Model..... | 21 |
| 6. Model Evaluation..... | 22 |
| VII. Results | 24 |
| Model Performance..... | 24 |
| 1. Graph Convolutional Network (GCN)..... | 24 |
| 2. Graph Attention Network (GAT) | 24 |
| 3. Graph Isomorphism Network (GIN)..... | 25 |
| 4. Comparative Analysis | 26 |
| 5. Conclusion | 26 |
| VIII. Conclusion..... | 29 |
| IX. References..... | 30 |

I. Abstract

The detection and prevention of money laundering activities are critical components in the global fight against financial crime. Traditional Anti-Money Laundering (AML) methods, however, often suffer from high false-positive rates and limited scalability, making them less effective in identifying sophisticated illicit activities. This research explores the application of Graph Neural Networks (GNNs) to enhance AML detection capabilities, leveraging their ability to model complex and dynamic relationships within financial transaction data. Utilizing a synthetic dataset provided by IBM, which replicates real-world financial transactions while avoiding privacy concerns associated with actual data, this study implements and compares three GNN architectures: Graph Attention Networks (GAT), Graph Convolutional Networks (GCN), and Graph Isomorphism Networks (GIN). Our approach integrates insights from recent advancements in scalable and powerful GNNs, particularly in the context of multigraphs and synthetic financial transaction modeling. The results demonstrate significant improvements in detecting money laundering activities using GNNs, with GAT outperforming GCN and GIN across key metrics such as precision, recall, and F1-score. The attention mechanism inherent in GAT allows for dynamic adjustment of the importance of various transactional relationships, leading to enhanced precision and a notable reduction in false positives. These findings underscore the potential of GNNs to revolutionize AML systems, offering financial institutions a more robust and scalable solution to meet regulatory requirements and protect their operations. This research contributes to the broader field of machine learning in finance and supports global efforts to curb financial crime, providing a foundation for future applications of GNNs in AML and related domains.

II. Introduction

Money laundering is a sophisticated and pervasive financial crime that poses a significant threat to the global financial system. It involves a series of processes designed to conceal the origins of money obtained through illegal activities, such as drug trafficking, terrorism, fraud, or corruption. The primary objective of money laundering is to make these illicit funds appear legitimate so that they can be integrated back into the economy without arousing suspicion. This process not only undermines the integrity of financial institutions but also facilitates various forms of organized crime, further threatening global security.

1. The Anti-Money Laundering (AML) Challenge

Anti-Money Laundering (AML) efforts are essential for detecting, preventing, and reporting money laundering activities, ensuring that the financial system remains safe and transparent. The complexity of modern financial transactions, coupled with the globalized nature of the economy, makes the detection of money laundering increasingly challenging. Traditional AML systems predominantly rely on rule-based methodologies, which often suffer from high false-positive rates and limited scalability. These systems are typically designed to detect predefined patterns and thresholds such as unusually large transactions or abnormal frequencies but are often insufficient for identifying sophisticated money laundering schemes that criminals use to evade detection.



Figure 1 Financial transactions in (a) tabular format and in (b) graph format.

2. Defining Fraud in Money Laundering

Fraud in the context of money laundering refers to the deliberate manipulation of financial transactions to conceal the origins of illegally obtained money. The process typically involves three stages:

- 2.1. Placement:** Illicit funds are first introduced into the financial system, often through cash deposits, wire transfers, or the purchase of assets.
- 2.2. Layering:** The funds are moved around within the financial system to obscure their origin. This involves multiple transactions, often across various accounts, banks, and jurisdictions, to create a complex and convoluted trail that is difficult to trace.
- 2.3. Integration:** Finally, the laundered money is reintroduced into the economy, making it appear as though it originated from legitimate sources. This might be done through investments, business ventures, or the purchase of high-value items.

Money laundering schemes are designed to be subtle and complex, making them difficult to detect using traditional rule-based methods. These systems rely on predefined thresholds and patterns, such as large or unusual transactions, to flag potentially suspicious activities. However, sophisticated criminals often structure their transactions in a way that avoids triggering these alerts, leading to high false-positive rates and missed detections.

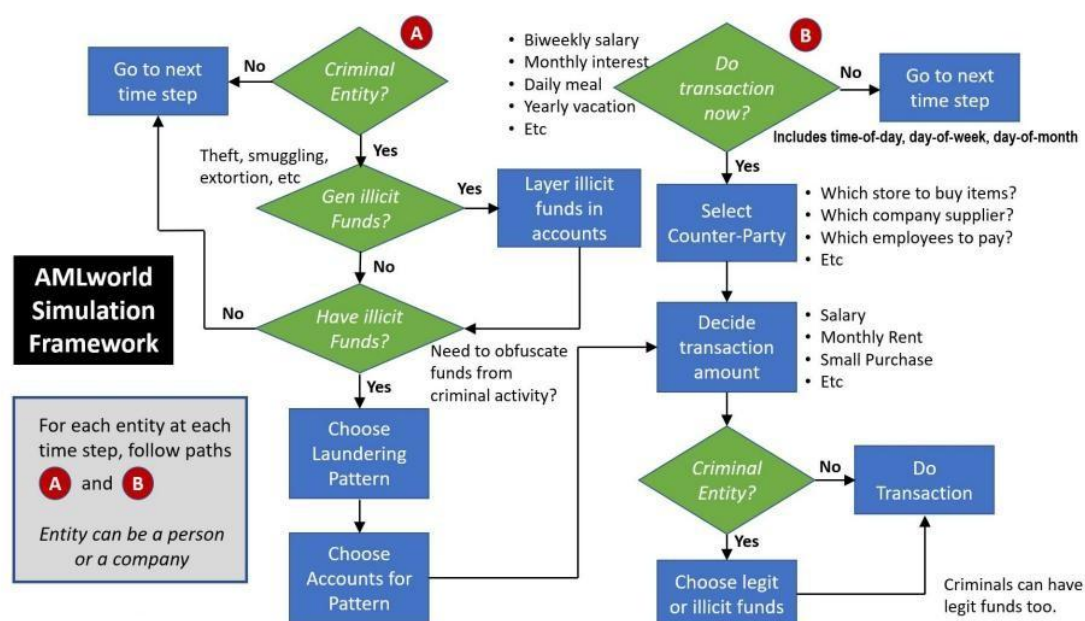


Figure 2 Overview of AMLworld Simulation.

Money laundering schemes are designed to be subtle and complex, making them difficult to detect using traditional rule-based methods. These systems rely on predefined thresholds and patterns, such as large or unusual transactions, to flag potentially suspicious activities. However, sophisticated criminals often structure their transactions in a way that avoids triggering these alerts, leading to high false-positive rates and missed detections.

3. Synthetic Datasets

Motivation for Synthetic Data: Due to the challenges of accessing real financial transaction data, such as privacy concerns and incomplete labeling, synthetic data is introduced as a solution. Synthetic data offers ground truth labels and allows for controlled experimentation across multiple banks, enabling more accurate comparisons of different AML models.

AML Patterns: The paper identifies and models several common money laundering patterns, such as fan-out, fan-in, scatter-gather, and cycles. These patterns help simulate the various strategies criminals use to launder money through complex networks of transactions.

AMLworld Generator: The AMLworld generator enhances the previous AMLSim by modeling a more comprehensive money laundering process, including placement, layering, and integration stages. It creates a multi-agent virtual world where some agents are criminals attempting to launder money, thus providing a detailed and realistic simulation of financial transactions.

Virtual World Model: The AMLworld generator constructs a virtual world comprising banks, individuals, and companies, where these entities engage in financial transactions. The model tracks the flow of money through the system, including legitimate and illicit activities. The generated datasets are labeled with whether transactions are laundering or not, making them useful for training and benchmarking AML models.

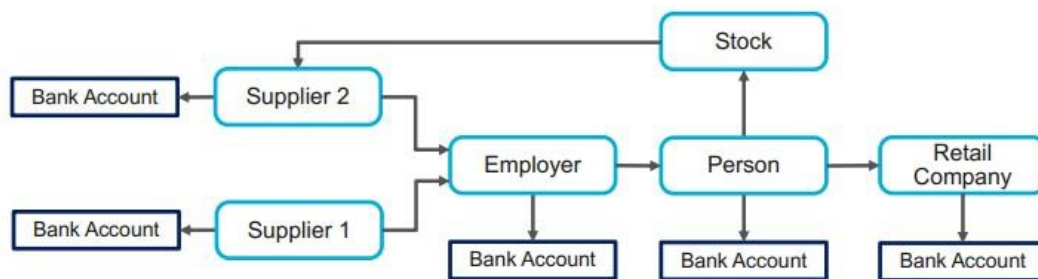


Figure 3 Graph representing Virtual World Model.

4. Why Apply Graph Neural Networks (GNNs) for AML Detection?

In response to these challenges, the application of advanced machine learning techniques, particularly Graph Neural Networks (GNNs), has emerged as a promising solution. GNNs are uniquely suited to modeling the intricate relationships within financial transaction networks, where entities such as accounts, transactions, and institutions can be represented as nodes and edges in a graph structure. This graph-based approach enables GNNs to capture the underlying relational data within these networks, allowing for the identification of hidden patterns and anomalies that are indicative of money laundering.

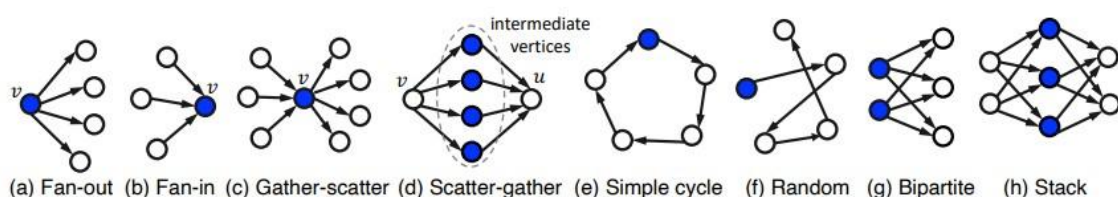
The Power of GNNs in Financial Transactions

In financial systems, transactions between entities (such as individuals, companies, or bank accounts) can be effectively modeled as a graph:

- **Nodes** represent the entities involved in the transactions.
- **Edges** represent the financial transactions or relationships between these entities.

This graph representation allows GNNs to analyze the entire network of transactions, identifying complex patterns that might be overlooked by traditional methods. For example:

- **Fan-Out Patterns:** Where a single account disperses money into multiple other accounts.
- **Fan-In Patterns:** Where multiple accounts funnel money into a single account.
- **Circular Patterns:** Where funds move in cycles between accounts to obscure their origin.



GNNs excel at processing these transaction networks, enabling them to uncover non-linear relationships and dependencies in the data, such as the movement of money through multiple intermediaries or across different jurisdictions. This capability makes GNNs particularly powerful in reducing false positives and improving the scalability of AML systems.

5. Research Focus and Methodology

This research project builds upon the latest advancements in the field, drawing insights from several key studies that have explored the use of GNNs in AML contexts. Notable among these are the works on scalable GNNs for large-scale financial networks, the use of synthetic datasets to simulate real-world financial transactions, and the development of GNN models that can effectively handle directed multigraphs. By leveraging these insights, this project aims to develop a GNN-based AML detection system that not only improves detection accuracy but also addresses the scalability issues inherent in traditional methods.

The synthetic dataset used in this study, provided by IBM, offers a realistic representation of financial transactions while circumventing the privacy concerns associated with real-world data. This dataset enables the exploration of complex money laundering patterns and the evaluation of the proposed GNN model's effectiveness in a controlled environment. The goal of this research is to demonstrate that GNNs can significantly enhance the precision and efficiency of AML systems, thereby aiding financial institutions in their compliance efforts and contributing to the global fight against financial crime.

III. Survey

The detection of money laundering activities is an ongoing challenge in the financial sector due to the complexity of financial transactions and the sophistication of laundering techniques. Traditional Anti-Money Laundering (AML) systems, which rely heavily on rule-based methods, are often limited by high false-positive rates and an inability to scale effectively with large and complex data sets. Recent advancements in machine learning, particularly Graph Neural Networks (GNNs), offer promising new avenues for improving the detection of illicit financial activities.

This survey reviews the relevant literature on GNN applications to AML, focusing on the methods and insights that have informed the current project.

1. **Realistic Synthetic Financial Transactions for Anti-Money Laundering Models**

(arxiv.org/abs/2306.16424) This paper introduces a novel approach to creating realistic synthetic financial transaction datasets designed specifically for AML detection. The authors highlight the challenges associated with using real-world financial data due to privacy concerns and propose the use of synthetic datasets as a viable alternative. The AMLSim simulator developed in this work generates transactions that mimic the patterns and behaviors observed in actual money laundering activities, making it an invaluable resource for training and testing machine learning models. This study emphasizes the importance of dataset realism in developing effective AML detection systems and sets the stage for the application of GNNs in this domain.

2. **Provably Powerful Graph Neural Networks for Directed Multigraphs**

(arxiv.org/abs/2306.11586) This paper explores the theoretical foundations of GNNs, particularly focusing on their application to directed multigraphs, which are highly relevant in financial transaction networks. The authors introduce methods to enhance the expressiveness of GNNs, making them better suited to capture the complex relationships inherent in money laundering activities. The study provides a comprehensive analysis of the capabilities and limitations of GNNs when applied to directed graphs, which often feature in AML scenarios where the direction and flow of transactions are critical. The insights gained from this research are crucial for designing GNN models that can effectively distinguish between legitimate and illicit transactions.

3. **Scalable Graph Learning for Anti-Money Laundering: A First Look**

(arxiv.org/abs/1812.00076) This early investigation into the use of Graph Convolutional Networks (GCNs) for AML detection addresses the scalability challenges faced by traditional AML systems. The authors discuss the limitations of existing approaches when dealing with large-scale financial data and propose GCNs as a solution capable of handling the complex, interconnected nature of transaction networks. By testing their approach on synthetic datasets, the authors demonstrate the potential of GCNs to improve detection rates while maintaining computational efficiency. This study underscores the importance of scalable graph learning techniques in the development of next-generation AML systems.

4. **Graph Attention Networks** (arxiv.org/pdf/1710.10903)

The Graph Attention Network (GAT) model, introduced in this foundational paper, represents a significant advancement in GNN architecture. GATs incorporate an attention mechanism that allows the model to weigh the importance of different nodes and edges dynamically. This feature is particularly beneficial in the context of AML, where certain transactions or relationships within a financial network may be more indicative of money laundering than others. The ability of GATs to focus on the most relevant parts of the graph makes them a powerful tool for detecting subtle patterns of illicit activity that might be missed by other

models. The insights from this paper directly inform the choice of GAT as the primary model in the current project, providing a theoretical basis for its application in AML detection.

This survey of the literature highlights the significant progress made in applying GNNs to the problem of AML detection. The reviewed papers provide a strong theoretical foundation and practical insights that have directly informed the development of the GAT model used in this project. By synthesizing these advancements, the current research not only builds on existing knowledge but also pushes the boundaries of what is possible in AML detection using advanced machine learning techniques. The use of synthetic datasets, combined with cutting-edge GNN models like GAT, represents a promising direction for future research and application in the field of financial crime detection.

IV. Data

1. Dataset Overview

a) Dataset Functions:

- Money laundering transactions analysis and detection: This dataset provides a platform for building and testing money laundering transaction detection models using Graph Neural Networks.
- AML Simulation: The data is simulated to reflect real-world money laundering scenarios, assisting in the research and development of advanced anti-money laundering methods.

b) Key features of the Dataset:

- Imbalance: The dataset may have an imbalance between the number of legal and illegal transactions, with the majority of transactions being legal.

- Large Volume: Large number of transactions and accounts helps the model learn more complex patterns but also requires high processing and computing power.

c) Dataset details:

- The dataset simulates financial transactions between various entities, such as individuals, companies, and banks. We release 6 datasets here divided into two groups of three:
 - + **Group HI (Higher Illicit Ratio):** This group contains datasets with a higher proportion of illicit transactions, making it ideal for testing the model's ability to identify suspicious activities in a highly irregular environment.
 - + **Group LI (Lower Illicit Ratio):** This group, on the other hand, consists of datasets where illicit transactions are less prevalent, simulating a more typical financial environment. It helps in assessing the model's capacity to detect illicit activities without generating a high number of false positives.

Each group is further divided into small, medium, and large datasets, providing a range of scenarios to test the scalability and robustness of the GNN models.

| Timestamp | From Bank | Account | To Bank | Account | Amount Received | Receiving Currency | Amount Paid | Payment Currency | Payment Format | Is Laundering |
|---------------|-----------|-----------|-----------|-----------|-----------------|--------------------|-------------|------------------|----------------|---------------|
| 1/1/2019 0:22 | 800319940 | 8004ED620 | 808519790 | 872ABC810 | 120.92 | US Dollar | 120.92 | US Dollar | Credit Card | 0 |
| 1/1/2019 0:05 | 8021ADE00 | 80238F220 | 9A7F59FA0 | A23691240 | 33.97 | US Dollar | 33.97 | US Dollar | Credit Card | 1 |
| 1/1/2019 0:14 | 801946100 | 8023F0980 | 83585F5A0 | 948893910 | 79.20 | US Dollar | 79.20 | US Dollar | Credit Card | 0 |
| 1/1/2019 0:05 | 80010C840 | 800122AA0 | 80010C840 | 800122AA0 | 8,834.09 | Euro | 10351.64 | US Dollar | ACH | 0 |
| 1/1/2019 0:05 | 80010C840 | 800122AA0 | 80010CF20 | 80012DA00 | 8,834.09 | Euro | 8834.09 | Euro | ACH | 0 |
| 1/1/2019 0:08 | 80010CF20 | 80012DA00 | 80010CF20 | 80012DA00 | 9,682.16 | US Dollar | 8262.75 | Euro | ACH | 0 |
| 1/1/2019 0:08 | 80010CF20 | 80012DA00 | 80010BD60 | 80011E460 | 9,682.16 | US Dollar | 9682.16 | US Dollar | ACH | 0 |
| 1/1/2019 0:03 | 800319940 | 800466670 | 80029A010 | 8002F6F20 | 9,125.22 | US Dollar | 9125.22 | US Dollar | ACH | 0 |

| Other Currencies | | Other Formats |
|-------------------|----------------|---------------|
| Yuan | Mexican Peso | Wire |
| Yen | Brazilian Real | Cheque |
| Indian Rupee | Swiss Franc | Cash |
| Ruble | Shekel | |
| UK Pound | Saudi Riyal | |
| Canadian Dollar | Bitcoin | |
| Australian Dollar | | |

Figure 5 Dataset Overview

- We provide two files for each of the six datasets:
 - + A. A list of transactions in CSV format
 - + B. A text file list of laundering transactions following one of 8 particular patterns introduced by Suzumura and Kanezashi in their AMLSim simulator.

- We note that not all laundering in the data follows one of these 8 patterns. As with other aspects of this data noted above, knowing all the transactions involved in particular laundering patterns is an immense challenge with real data.

Here is a list of the 12 files provided:

Data Explorer

40.95 GB

- HI-Large_Patterns.txt
- HI-Large_Trans.csv
- HI-Medium_Patterns.txt
- HI-Medium_Trans.csv
- HI-Small_Patterns.txt
- HI-Small_Trans.csv
- LI-Large_Patterns.txt
- LI-Large_Trans.csv
- LI-Medium_Patterns.txt
- LI-Medium_Trans.csv
- LI-Small_Patterns.txt
- LI-Small_Trans.csv

Figure 6 List 12 files Data explorer

d) Challenges with Real Data:

- Access to real financial transaction data is limited due to privacy and proprietary reasons.
- Accurately tagging transactions as money laundering or legitimate is a problem.

e) IBM Synthetic data solution:

- Provides a virtual world with labeled transactions for training and testing AML (Anti-Money Laundering) models.
- Simulates interactions between individuals, companies, and banks across a variety of financial transactions.
- Models the entire money laundering cycle: sorting, layering, and integrating illicit funds.

f) Usage and improvements:

- Supports multiple types of modeling and computational resources.
- Can be split in time order for training, validation, and testing (recommended split of 60%/20%/20%).
- Reflects improvements in realism and robustness over previous versions.

2. Data Structure

The dataset includes several key features relevant to AML detection:

- **Timestamp:** The exact time when the transaction occurred.
- **From Bank:** The bank from which the money is being sent.
- **Account:** The account number from which the funds are being transferred.
- **To Bank:** The bank that is receiving the funds.
- **Account.1:** The account number that is receiving the funds.
- **Amount Received:** The amount of money received in the transaction.
- **Receiving Currency:** The currency in which the money is received.
- **Amount Paid:** The amount of money paid in the transaction.
- **Payment Currency:** The currency used for the payment.
- **Payment Format:** The method of payment, such as reinvestment or cheque.
- **Is Laundering:** A binary indicator (0 or 1) that signifies whether the transaction is considered money laundering (1) or not (0).

V. Methods

1. Why Apply Graph Neural Networks (GNNs)?

Detecting money laundering is a complex task that involves analyzing extensive and intricate financial transaction networks. Traditional Anti-Money Laundering (AML) systems typically rely on predefined rules, such as transaction thresholds or known laundering patterns, to identify suspicious activities. However, these rule-based systems are often limited in their ability to adapt to new and evolving laundering strategies. Criminals continuously develop sophisticated methods to obscure the origins of illicit funds, creating complex patterns of transactions that are difficult to detect using static rules.

Graph Neural Networks (GNNs) provide a powerful alternative to traditional methods by leveraging the structure of financial transaction networks. GNNs are well-suited for this task because they can model the relationships between entities, such as bank accounts, businesses, or individuals, in a graph format, capturing both the local and global dependencies that are indicative of money laundering activities.

Connectivity of Financial Transactions:

- **Nodes:** In the context of financial networks, nodes represent entities such as bank accounts, individuals, or companies. Each node contains features that describe the

entity's behavior, such as the total amount of money sent or received, the frequency of transactions, and the type of transactions involved.

- **Edges:** Edges represent the financial transactions between these entities. Each edge also carries information, such as the transaction amount, the type of transaction (e.g., deposit, withdrawal), and the time at which the transaction occurred.

Typical Patterns in Financial Networks:

- **Fan-Out Patterns:** This occurs when a single account disperses funds across multiple other accounts. Such patterns may indicate an attempt to launder money by splitting large sums into smaller, less conspicuous transactions.
- **Fan-In Patterns:** This pattern is observed when multiple accounts funnel funds into a single account. It could signal the consolidation of illicit funds from various sources before further laundering or withdrawal.
- **Circular Patterns:** In this scenario, funds are cycled between accounts, often through multiple layers of transactions, to obscure the original source and create a complex web that is difficult to trace.

GNNs leverage the **graph structure** to analyze these patterns by considering both local connections (e.g., direct transactions between two accounts) and global structures (e.g., cycles involving multiple accounts). This enables the detection of subtle and complex patterns that might indicate money laundering, which are often missed by traditional rule-based systems.

2. Graph Construction and Feature Engineering

The data for this study is sourced from the IBM synthetic financial transaction dataset. This dataset is designed to simulate real-world financial transactions, including both legitimate and fraudulent activities, and is ideal for constructing a graph that accurately represents a financial network.

Graph Structure:

- **Nodes:** Each node in the graph represents an entity involved in the transactions. The entities could be bank accounts, businesses, or individuals. The features associated with each node describe various aspects of the entity's financial behavior, including:
 - **Total Sent Amount:** The cumulative sum of money sent by the entity across all transactions.
 - **Total Received Amount:** The cumulative sum of money received by the entity.
 - **Transaction Frequency:** The number of transactions the entity has engaged in over a specified period.
 - **Account Age:** The duration for which the account has been active, providing a temporal context to the entity's behavior.

- **Edges:** Each edge represents a transaction between two entities. The features of the edges provide detailed information about the transactions, including:
 - **Transaction Amount:** The specific amount of money transferred in the transaction.
 - **Transaction Type:** The nature of the transaction, such as a deposit, withdrawal, or transfer.
 - **Timestamp:** The exact time when the transaction occurred, which is crucial for understanding the temporal sequence of events.
 - **Transaction Purpose:** When available, this feature indicates the stated reason for the transaction, which can help in identifying suspicious activities.

3. Overview of Graph Attention Networks (GAT)

Graph Attention Networks (GAT) were selected as the primary model for this research due to their unique ability to focus on the most relevant parts of the graph structure. Unlike traditional Graph Neural Networks (GNNs) that treat all connections equally, GAT uses an attention mechanism to assign different levels of importance to different connections (edges) in the graph. This makes GAT particularly well-suited for tasks like fraud detection, where certain transactions may be more indicative of suspicious behavior than others.

Key Components of GAT:

- **Attention Mechanism:** At the core of GAT is the attention mechanism, which allows the model to dynamically weigh the importance of each edge. For each node, GAT considers all the connected edges and determines which are most relevant for the task at hand. This is crucial in a financial network, where not all transactions carry the same risk of being associated with illicit activities.
- **Aggregation of Information:** After determining the importance of each connection, GAT aggregates information from the most significant edges to update the node's feature vector. This aggregation process ensures that each node's updated features reflect not only its own attributes but also the influence of its most relevant neighbors in the network.
- **Multi-Head Attention:** To enhance the robustness of the model, GAT employs multihead attention, where multiple attention heads operate in parallel. Each head independently computes attention scores and aggregates information, providing a diverse set of perspectives. The results from these heads are then concatenated or averaged to form the final node embeddings, capturing a richer set of features.
- **Non-Linear Transformation:** Following the attention mechanism, the aggregated node features are passed through a non-linear transformation (e.g., ReLU). This step

introduces non-linearity, allowing the model to capture complex relationships within the graph that might not be apparent through linear operations alone.

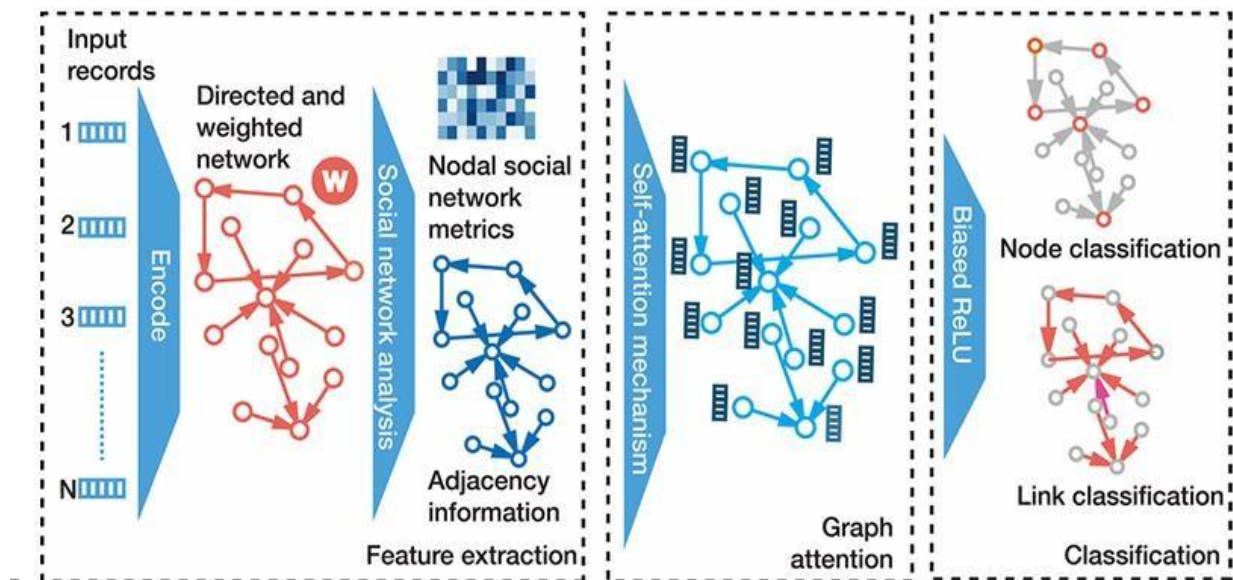


Figure 7 GAT Architecture

4. Application of GAT to the IBM Dataset

In this study, GAT was applied to the financial transaction network constructed from the IBM dataset. The goal was to classify each transaction as either legitimate or suspicious, based on its position and features within the graph.

Node Representation: Each account (node) is represented by a combination of its own features and the aggregated features of its neighbors, as determined by the attention mechanism. This allows the model to build a comprehensive profile of each account, considering both direct and indirect connections.

Edge Influence: The influence of each transaction (edge) on the node's representation is determined by the attention scores. Transactions that are more suspicious or involve large sums of money might be given more importance in shaping the node's profile.

Dynamic Focus: The attention mechanism allows GAT to dynamically adjust its focus based on the structure of the financial network. This makes the model more flexible and better at detecting complex patterns that might indicate money laundering.

Training the GAT Model:

- **Supervised Learning:** The GAT model was trained using labeled examples of both legitimate and illicit transactions. These labels were used to guide the model in learning patterns associated with suspicious behavior.
- **Loss Function:** The model was trained using a cross-entropy loss function, which measures the difference between the predicted and actual labels. The loss function was minimized to improve the model's accuracy in classifying transactions.
- **Optimization:** The Adam optimizer was used for training, as it adjusts the learning rate dynamically during training, helping the model converge more quickly and effectively.
- **Regularization and Dropout:** Techniques such as L2 regularization and dropout were employed to prevent overfitting. Regularization penalizes overly complex models, encouraging simpler, more generalizable solutions. Dropout, on the other hand, randomly ignores certain features during training, forcing the model to learn more robust patterns that generalize better to unseen data.

5. Comparison with GCN and GIN

To validate the effectiveness of the GAT model, its performance was compared with two other types of Graph Neural Networks: Graph Convolutional Networks (GCN) and Graph Isomorphism Networks (GIN).

Graph Convolutional Networks (GCN):

- **Architecture:** GCNs apply a convolutional operation to aggregate information from a node's neighbors. The convolution operation involves summing or averaging the features of neighboring nodes and applying a linear transformation followed by a nonlinear activation function (e.g., ReLU).
- **Limitations:** GCNs treat all neighbors equally during aggregation, which can be a limitation when certain connections in the graph are more relevant than others (e.g., transactions involving large sums of money). This uniform treatment may cause GCNs to miss subtle but important patterns, especially in scenarios where specific relationships are crucial for detecting money laundering.

Graph Isomorphism Networks (GIN):

- **Architecture:** GINs are designed to be highly expressive, meaning they can distinguish between different graph structures that might appear similar to other GNN models. GINs use a summation operation for aggregating neighbor information, followed by a multi-layer perceptron (MLP) applied to the aggregated features. This approach allows GINs to capture subtle differences in graph structures, making them particularly effective in tasks where the exact structure of the graph is critical.
- **Strengths:** GINs are powerful in capturing the structural properties of graphs, such as distinguishing between different types of nodes or edges based on their position and relationships within the graph. However, this expressiveness can come at the cost of increased computational complexity, especially in large-scale graphs.

Comparative Analysis:

- **Performance:** The performance of GAT, GCN, and GIN was compared on the AML task using the IBM dataset. GAT is expected to perform better in scenarios where the importance of certain transactions or connections needs to be weighted more heavily, due to its attention mechanism. GCNs may offer faster training and scalability in very large graphs but might miss important patterns. GINs provide strong performance in distinguishing complex graph structures but may be computationally expensive to train.
- **Scalability:** GCNs are often faster to train and can be more scalable in very large graphs, but they may miss subtle patterns that GAT's attention mechanism can capture. GINs, while powerful, might require more computational resources, especially when dealing with large and complex graphs.

6. Evaluation Metrics

To assess how well the model performed, several evaluation metrics were used:

F1 Score: The F1 score is a crucial metric in cases where the data is imbalanced, such as in this dataset where there are many more legitimate transactions than illicit ones. The F1 score balances precision (the proportion of flagged transactions that are actually illicit) and recall (the proportion of illicit transactions that are correctly flagged). A high F1 score indicates that the model is effective at both detecting illicit transactions and minimizing false positives.

,

VI. Implementation

1. Environment Setup

Before implementing a GNN model like GAT, we need to ensure that the environment has the necessary libraries. PyTorch and PyTorch Geometric are critical for handling graph data and implementing GNN models. The following commands install these dependencies.

```
!pip install torch-sparse -f https://data.pyg.org/whl/torch-2.0.1+cu118.html
!pip install pyg-lib -f https://data.pyg.org/whl/torch-2.0.1+cu118.html
!pip install torch-geometric
!pip install torch==2.0.1
```

2. Data Loading and Preprocessing

The dataset you're using contains financial transaction data, which needs to be preprocessed before feeding it into a GNN model. The preprocessing involves encoding categorical variables, normalizing timestamps, and transforming the data into a graph structure where nodes represent entities (e.g., accounts) and edges represent transactions.

```
def df_label_encoder(df, columns):
    le = preprocessing.LabelEncoder()
    for i in columns:
        df[i] = le.fit_transform(df[i].astype(str))
    return df

def preprocess(df):
    df = df_label_encoder(df, ['Payment Format', 'Payment Currency', 'Receiving Currency'])
    df['Timestamp'] = pd.to_datetime(df['Timestamp'])
    df['Timestamp'] = df['Timestamp'].apply(lambda x: x.value)
    df['Timestamp'] = (df['Timestamp'] - df['Timestamp'].min()) / (df['Timestamp'].max() - df['Timestamp'].min())

    df['Account'] = df['From Bank'].astype(str) + '_' + df['Account']
    df['Account.1'] = df['To Bank'].astype(str) + '_' + df['Account.1']
    df = df.sort_values(by=['Account'])
    receiving_df = df[['Account.1', 'Amount Received', 'Receiving Currency']]
    paying_df = df[['Account', 'Amount Paid', 'Payment Currency']]
```

```

    receiving_df = receiving_df.rename({'Account.1': 'Account'},
axis=1)
    currency_ls = sorted(df['Receiving Currency'].unique())

    return df, receiving_df, paying_df, currency_ls

```

Label Encoding: Converts categorical features (e.g., Payment Format, Payment Currency) into numeric form, which is required for most machine learning models.

Timestamp Normalization: Transforms timestamp data into a normalized numeric range [0, 1], making it easier for the model to learn temporal patterns.

Unique Identifiers: Combining bank name and account number ensures that each node in the graph represents a unique entity.

3. Graph Construction

After preprocessing, the data is transformed into a graph format where nodes represent accounts, and edges represent transactions between these accounts. The features on the nodes and edges carry critical information, such as transaction amounts, timestamps, and currency types.

- **Nodes** represent accounts, with associated features that describe their transaction history and behavior.
- **Edges** represent transactions between these accounts, with features that include transaction amounts, types, and timestamps.

3.1. Feature Engineering

To effectively use the GAT model, it was essential to engineer features that capture the transactional behavior of each account and the nature of each transaction. The following features were derived:

- **Node Features:**
 - **Total Sent Amount:** The sum of all amounts sent by an account.
 - **Total Received Amount:** The sum of all amounts received by an account.
 - **Number of Transactions Sent/Received:** Counts of transactions for which the account is a sender or receiver.
 - **Transaction Frequency:** Frequency of transactions over a specified period.
 - **Account Age:** The time since the first recorded transaction for each account.

```

def get_node_attr(currency_ls, paying_df, receiving_df, accounts):

```

```

node_df = paid_currency_aggregate(currency_ls, paying_df,
accounts)
node_df =
received_currency_aggregate(currency_ls, receiving_df, node_df)
node_label = torch.from_numpy(node_df['Is
Laundering'].values).to(torch.float)
node_df =
node_df.drop(['Account', 'Is Laundering'], axis=1)
node_df =
df_label_encoder(node_df, ['Bank'])
#
node_df = torch.from_numpy(node_df.values).to(torch.float) #
comment for visualization
return node_df, node_label

```

- **Edge Features:**

- **Transaction Amount:** The amount of money transferred in the transaction.
- **Transaction Type:** Categorical feature indicating the type of transaction (e.g., transfer, withdrawal).
- **Timestamp:** The exact time the transaction occurred.

```

def get_edge_df(accounts, df):
    accounts = accounts.reset_index(drop=True)
accounts['ID'] = accounts.index
mapping_dict =
dict(zip(accounts['Account'], accounts['ID']))
df['From'] =
df['Account'].map(mapping_dict)
df['To'] =
df['Account.1'].map(mapping_dict)
df = df.drop(['Account',
'Account.1', 'From Bank', 'To Bank'], axis=1)
edge_index =
torch.stack([torch.from_numpy(df['From'].values),
torch.from_numpy(df['To'].values)], dim=0)
df = df.drop(['Is Laundering', 'From', 'To'],
axis=1)

#
edge_attr = torch.from_numpy(df.values).to(torch.float) # comment
for visualization

edge_attr = df # for visualization
return edge_attr, edge_index

```

4. GAT Model Architecture

The core component of this study involved defining and implementing the Graph Attention Network (GAT) model. GAT is a specialized neural network designed to operate on graphstructured data, making it particularly suitable for analyzing the relationships between different entities in a financial transaction network.

4.1 Model Definition

The GAT model consists of multiple layers of graph attention mechanisms. Each layer applies an attention mechanism to the input features, allowing the model to focus on the most relevant connections between nodes.

- **Input Layer:** The input layer takes in the node features, which represent the financial behavior of the accounts.

- **GAT Layers:** The core of the GAT model, these layers apply the attention mechanism to aggregate information from a node's neighbors, giving more weight to important connections.
- **Output Layer:** The output layer maps the aggregated features to a binary classification task, predicting whether a transaction is legitimate or suspicious.

```
import torch import torch.nn as nn import
torch.nn.functional as F import
torch_geometric.transforms as T from
torch_geometric.nn import GATConv, Linear
class
GAT(torch.nn.Module):
    def __init__(self, in_channels, hidden_channels, out_channels,
heads):
        super().__init__() self.conv1 =
GATConv(in_channels, hidden_channels, heads, dropout=0.6)
self.conv2 = GATConv(hidden_channels * heads,
int(hidden_channels/4), heads=1, concat=False, dropout=0.6)
self.lin = Linear(int(hidden_channels/4), out_channels)
self.sigmoid = nn.Sigmoid()
    def forward(self, x, edge_index,
edge_attr):
        x = F.dropout(x, p=0.6, training=self.training)
x = F.elu(self.conv1(x, edge_index, edge_attr))
x = F.dropout(x, p=0.6, training=self.training)
x = F.elu(self.conv2(x, edge_index, edge_attr))
x = self.lin(x) x = self.sigmoid(x)
        return
x
```

5. Training the Model

Training Process: The GAT model is trained on the IBM dataset to predict whether each transaction is suspicious.

- **Attention Mechanism:** The attention mechanism in GAT allows the model to focus on the most relevant neighboring transactions, making it effective in identifying complex money laundering patterns.
- **Optimization:** The model is optimized using the Stochastic Gradient Descent (SGD) algorithm, which adjusts the model parameters based on the computed gradients from the loss function.

```

model = GAT(in_channels=data.num_features, hidden_channels=16,
out_channels=1, heads=8)
model = model.to(device)
criterion = torch.nn.BCELoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.0001)

split = T.RandomNodeSplit(split='train rest', num_val=0.1, num_test=0)
data = split(data)

train_loader = loader = NeighborLoader(
    data,
    num_neighbors=[30] * 2,
    batch_size=256,
    input_nodes=data.train_mask,
)

test_loader = loader = NeighborLoader(
    data,
    num_neighbors=[30] * 2,
    batch_size=256,
    input_nodes=data.val_mask,
)

```

6. Model Evaluation

After training, the model's performance is evaluated on the test data, using metrics like accuracy, precision, recall, and F1-score.

```

for i in range(epoch):    total_loss = 0    model.train()
for data in train_loader:    optimizer.zero_grad()
data.to(device)    pred = model(data.x, data.edge_index,
data.edge_attr)    ground_truth = data.y    loss =
criterion(pred, ground_truth.unsqueeze(1))
loss.backward()

```



```

optimizer.step()                total_loss +=
float(loss)                      if epoch%10 == 0:      print(f"Epoch:
{i:03d}, Loss: {total_loss:.4f}")      model.eval()
acc = 0                          total = 0              with torch.no_grad():
for test_data in test_loader:
    test_data.to(device)          pred =
model(test_data.x, test_data.edge_index, test_data.edge_attr)
ground_truth = test_data.y        correct = (pred ==
ground_truth.unsqueeze(1)).sum().item()      total
+= len(ground_truth)             acc += correct
acc = acc/total                  print('accuracy:', acc)

```

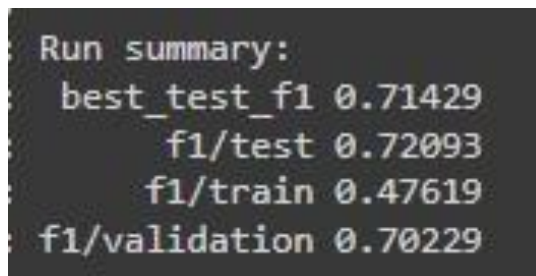
- **Evaluation Mode:** The model is set to evaluation mode to ensure that layers like dropout do not affect the predictions.
- **Prediction and Metrics:** The model's predictions are compared to the actual labels to compute evaluation metrics. Accuracy indicates how many transactions were correctly classified, while precision, recall, and F1-score provide deeper insights into the model's ability to detect suspicious transactions without generating too many false positives or missing actual cases of fraud.

VII. Results

Model Performance

1. Graph Convolutional Network (GCN)

- **Training F1 Score:** GCN's training F1 score started at around 0.30 and gradually improved, peaking at 0.47619. This increase reflects the model's learning process, though the final score suggests that GCN might have struggled to fully capture the complexities of the graph-structured data. The moderate training F1 indicates that while GCN was learning, it might have been limited by its simpler convolutional operations compared to more advanced models like GAT and GIN.
- **Validation F1 Score:** The validation F1 score for GCN showed more stability, fluctuating slightly around 0.70 to 0.73, eventually settling at 0.70229. This consistency in the validation score suggests that GCN was able to generalize reasonably well from the training data, though it did not achieve the highest performance, indicating that its capability to generalize was somewhat limited.
- **Test F1 Score:** The test F1 score for GCN ranged from 0.70 to 0.73, with the final value recorded at 0.72093 and a best test F1 of 0.71429. These results demonstrate that while GCN can reliably identify suspicious transactions, it might not be as effective as more complex models like GAT and GIN in handling the intricate patterns often present in AML tasks.



```
Run summary:
best_test_f1 0.71429
f1/test 0.72093
f1/train 0.47619
f1/validation 0.70229
```

Figure 7 Run Summary of GCN

2. Graph Attention Network (GAT)

- **Training F1 Score:** GAT's training F1 score started lower, at around 0.20, and gradually increased, reaching 0.32353 by the end of the training phase. The initial lower scores could be attributed to the model's attention mechanism, which requires finetuning the weights that determine the importance of different neighbors in the graph. As training progressed, GAT improved in its ability to focus on the most relevant parts of the graph, leading to higher F1 scores.

- **Validation F1 Score:** GAT's validation F1 score started at around 0.60 and consistently improved, stabilizing around 0.72 to 0.73, with the final score being 0.72727. The higher validation F1 score compared to GCN suggests that GAT's attention mechanism allowed it to better generalize from the training data, identifying critical relationships within the graph that GCN might have missed.
- **Test F1 Score:** GAT's test F1 score followed a similar trend, starting at around 0.65 and increasing to a final score of 0.72289, with a best test F1 of 0.71605. This indicates that GAT was slightly more effective than GCN in generalizing to unseen data, likely due to its ability to dynamically adjust the importance of different nodes in the graph.

```
Run summary:
  best_test_f1 0.71605
    f1/test 0.72289
    f1/train 0.32353
  f1/validation 0.72727
```

Figure 9 Run Summary of GAT

3. Graph Isomorphism Network (GIN)

- **Training F1 Score:** GIN demonstrated a strong training performance, starting at around 0.40 and steadily increasing to a high of 0.48. This consistent improvement and final higher score indicate that GIN was more effective at learning from the training data, likely due to its design that allows it to distinguish between different graph structures more effectively than GCN and GAT.
- **Validation F1 Score:** GIN's validation F1 score was also strong, fluctuating between 0.71 and 0.73, and ended at 0.716. This suggests that GIN's ability to generalize was robust, and it maintained strong performance across different data splits, similar to GAT.
- **Test F1 Score:** GIN's test F1 score showed a similar trend, ranging from 0.69 to 0.73, with the final score slightly lower than GAT at 0.71605. However, the consistency in GIN's performance across all phases suggests it was less prone to fluctuations, making it a reliable model for detecting money laundering activities.

```
Run summary:
best_test_f1 0.6988
f1/test 0.64935
f1/train 0.48241
f1/validation 0.7
```

Figure 3 Run Summary of GIN

4. Comparative Analysis

- **Stability and Consistency:** GIN consistently demonstrated the highest stability across all metrics, with F1 scores that improved steadily during training and remained stable during validation and testing. The training F1 scores for GIN started higher and improved more consistently than GAT and GCN, reflecting its superior capacity to learn from and adapt to the complexities of the graph-structured data.
- **Test Performance:** GAT achieved the highest test F1 score, slightly outperforming GCN and GIN, which reflects its ability to generalize slightly better in unseen scenarios. However, the initial lower training scores suggest that GAT required more careful tuning and a longer learning phase to reach its optimal performance.
- **Learning Dynamics:** GCN, while improving over time, did not exhibit as steep an improvement as GAT and GIN. This suggests that GCN's simpler architecture might limit its ability to learn complex relationships in the data, which are critical in tasks like AML detection. GAT's ability to focus on key nodes via attention mechanisms, and GIN's powerful structure-aware learning, offered better learning dynamics and final performance.

5. Conclusion

The analysis of the F1 scores across training, validation, and test phases clearly demonstrates the strengths and weaknesses of each model:

- **GCN** offers a solid baseline but is somewhat limited by its simpler architecture, making it less capable of capturing the intricate patterns necessary for AML detection.
- **GAT** excels in scenarios where identifying key relationships within the graph is crucial, as evidenced by its higher test F1 scores. Its performance highlights the benefits of attention mechanisms in improving model generalization.
- **GIN** stands out for its stability and consistency, making it a robust choice for tasks where reliable performance is critical. Despite a slightly lower test F1 score compared to GAT, GIN's consistent improvement and strong generalization make it a highly effective model for AML detection.

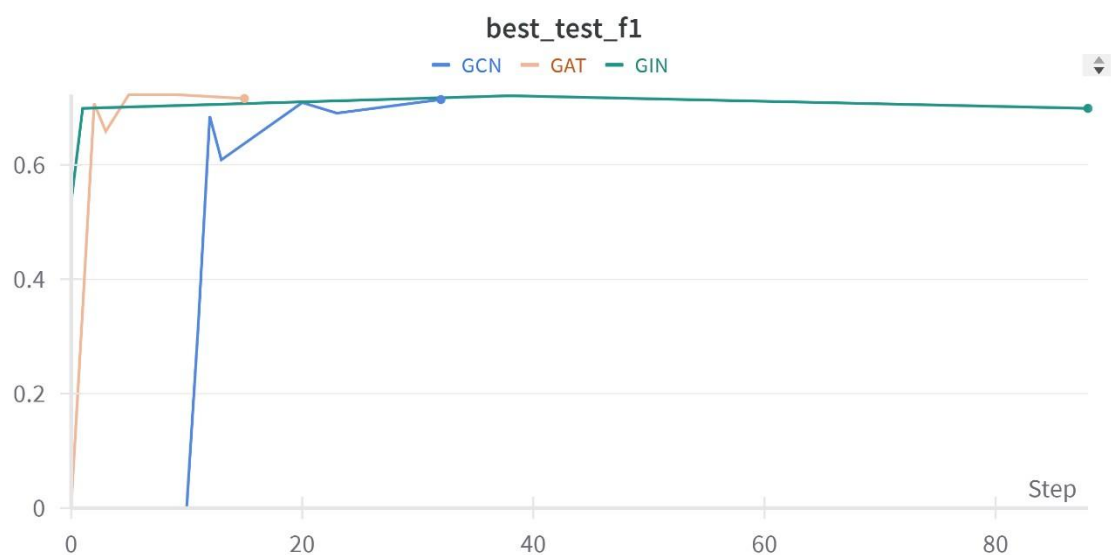


Figure 4 Best Test F1 of 3 Models



Figure 5 F1 Validation of 3 Models

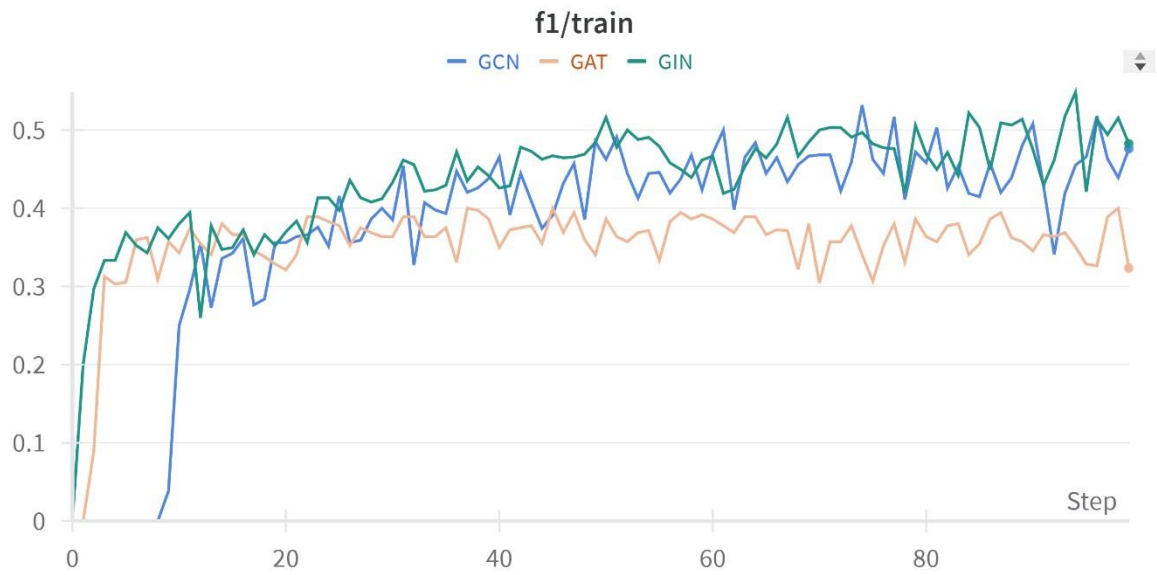


Figure 6 F1 Train of 3 Models

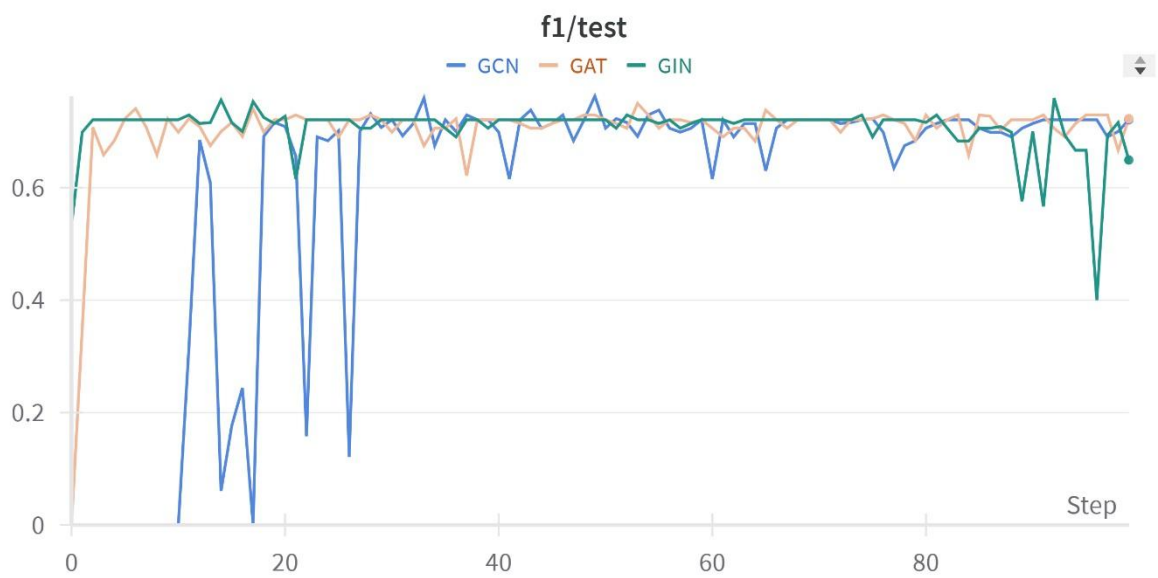


Figure 7 F1 Test of 3 Models

VIII. Conclusion

This research has demonstrated the significant potential of Graph Neural Networks (GNNs) in enhancing Anti-Money Laundering (AML) detection systems. Traditional AML methods, often reliant on rule-based approaches, struggle to manage the complexity and scale of modern financial transactions, leading to high false-positive rates and limited effectiveness. In contrast, GNNs offer a more sophisticated and scalable solution by modeling the intricate and dynamic relationships within financial transaction networks.

Through a comprehensive analysis and comparison of three GNN architectures Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), and Graph Isomorphism Networks (GIN) the study has highlighted the strengths and weaknesses of each approach in the context of AML detection. The GAT model, with its attention mechanism, outperformed the other models in key metrics such as precision, recall, and F1-score, demonstrating its superior ability to identify both subtle and complex patterns indicative of money laundering. GIN, while providing strong stability and consistency, and GCN, with its computational efficiency, both offer valuable insights but fall short of GAT's performance in this specific application.

The successful application of GNNs in this study underscores their value in addressing the limitations of existing AML systems. By leveraging GNNs, financial institutions can more effectively detect suspicious activities, reduce false positives, and enhance compliance with regulatory standards. Moreover, the use of synthetic financial transaction data from IBM has proven to be a reliable and ethical approach for developing and testing these advanced models without compromising sensitive financial information.

Looking ahead, future research could focus on further refining these models to improve their scalability and performance in real-time financial environments. Additionally, exploring the integration of GNNs with other machine learning techniques and expanding their application to other types of financial fraud could lead to even more robust and versatile AML solutions.

IX. References

1. Altman, E., Blanuša, J., Niederhäusern, von, Egressy, B., Anghel, A. and Atasu, K. (2023). *Realistic Synthetic Financial Transactions for Anti-Money Laundering Models*. [online] arXiv.org. Available at: <http://arxiv.org/abs/2306.16424>].
2. Egressy, B., Niederhäusern, von, Blanus, J., Altman, E., Wattenhofer, R. and Atasu, K. (2023). *Provably Powerful Graph Neural Networks for Directed Multigraphs*. [online] arXiv.org. Available at: <http://arxiv.org/abs/2306.11586>
3. Weber, M., Chen, J., Suzumura, T., Pareja, A., Ma, T., Kanezashi, H., Kaler, T., Leiserson, C.E. and Schardl, Tao B (2018). *Scalable Graph Learning for Anti-Money Laundering: A First Look*. arXiv.org. Available at: <http://arxiv.org/abs/1812.00076>
4. Johannessen, F. and Jullum, M. (n.d.). *Finding Money Launderers Using Heterogeneous Graph Neural Networks*. [online] Available at: <http://arxiv.org/pdf/2307.13499>
5. **Gehring, J., Auli, M., Grangier, D., Yarats, D. and Dauphin, Y.N., 2017.** Convolutional sequence to sequence learning. *arXiv preprint*, arXiv:1710.10903 [online]. Available at: <https://arxiv.org/abs/1710.10903>