Banking Fraud Detection Using GNN

Thesis to be submitted in partial fulfillment of the requirements for the degree

of

Master of Technology

by

Priyank Mundra
20EC39024

Under the guidance of

Prof. Pabitra Mitra and Prof. Arijit De

ELECTRONICS AND ELECTRICAL COMMUNICATION ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Department of Electronics and Electrical
Communication Engineering
Indian Institute of Technology, Kharagpur
India - 721302

CERTIFICATE

This is to certify that we have examined the thesis entitled **Banking Fraud Detection Using GNN**, submitted by **Priyank Mundra** (Roll Number: *20EC39024*) an undergraduate student of **Department of Electronics and Electrical Communication Engineering** in partial fulfillment for the award of degree of Master of Technology. We hereby approve it as a study carried out and presented in a manner required for its acceptance in partial fulfillment of the postgraduate degree for which it has been submitted. The thesis has met all the requirements according to the Institute regulations and reached the required submission standard.

Pabitra Mitra

Co-Supervisor
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Supervisor
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Date: 29th April 2025
Place: Kharagpur

ACKNOWLEDGEMENTS

I express my profound gratitude to all those who have contributed to the completion of this thesis. First and foremost, I would like to extend my heartfelt thanks to my supervisor, Prof. Pabitra Mitra, and my co-supervisor Prof. Arijit De for their unwavering support, insightful guidance, and invaluable expertise throughout this research journey. Their dedication and encouragement played a pivotal role in shaping the direction of this study.

I am deeply grateful to the faculty members of the Indian Institute of Technology, Kharagpur, whose knowledge and feedback enriched my understanding and inspired me to delve deeper into the intricate domain of Deep Learning.

My sincere appreciation goes to my colleagues and friends who provided endless encouragement, engaging discussions, and moral support. Your camaraderie made this academic endeavor a truly enriching experience. I am indebted to my family for their unconditional love, understanding, and encouragement. Their unwavering belief in my abilities kept me motivated during challenging times.

Lastly, I extend my gratitude to the academic community, researchers, and authors whose work laid the foundation for this study. Your contributions have been invaluable in shaping my perspective and improving the quality of this research.
This thesis stands as a testament to the collective support and inspiration I have received, and I am profoundly grateful for each individual who has been part of this academic endeavor.

Priyank Mundra
Indian Institute of Technology, Kharagpur
Date: 29th April 2025

ABSTRACT

Financial fraud poses a significant challenge due to the increasing complexity of transaction networks. This thesis explores the application of **Graph Neural Networks (GNNs)** to enhance fraud detection by modeling financial transactions as graphs, where accounts are nodes and transactions are edges. Specifically, we investigate and compare **Graph Attention Networks (GATs)** and **Spatial-Temporal Graph Attention Networks (STGATs)**.

The study addresses critical challenges like severe data imbalance and feature heterogeneity through careful preprocessing and distinct graph construction strategies: a homogeneous graph for **GAT** and a heterogeneous graph incorporating temporal dynamics for **STGAT**. Comparative analysis revealed that **STGAT**, by leveraging temporal information and heterogeneous structures, demonstrated significantly superior performance in identifying fraudulent transactions compared to **GAT**. Furthermore, the **STGAT** model proved effective and scalable when evaluated on the complete, large-scale dataset, maintaining robust fraud detection capabilities.

The findings underscore the substantial potential of advanced **GNNs**, particularly spatio-temporal models, for developing robust and scalable financial fraud detection systems. Future work includes exploring more sophisticated architectures, subgraph-level classification to detect complex fraud trails, generative models for transaction path reconstruction, improving model interpretability, and developing real-time adaptive learning capabilities.

Contents

List of Figures

List of Abbreviations

AML	**A**nti-**M**oney **L**aundering
CNN	**C**onvolutional **N**eural **N**etwork
ConvGNN	**C**onvolutional **G**raph **N**eural **N**etwork
EGAT	**E**dge-**F**eatured **G**raph **A**ttention **N**etwork
GAT	**G**raph **A**ttention **N**etwork
GCN	**G**raph **C**onvolutional **N**etwork
GIN	**G**raph **I**somorphism **N**etwork
GNN	**G**raph **N**eural **N**etwork
GraphSAGE	**G**raph **SA**mple and aggre**G**at**E**
MLP	**M**ulti-**L**ayer **P**erceptron
RNN	**R**ecurrent **N**eural **N**etwork
STGAT	**S**patial-**T**emporal **G**raph **A**ttention **N**etwork
STGNN	**S**patial-**T**emporal **G**raph **N**eural **N**etwork
TPR	**T**rue **P**ositive **R**ate

Chapter 1

Introduction

1.1 Overview of Financial Fraud and Money Laundering

Financial fraud and money laundering represent significant threats to the global economy, costing billions annually and undermining the integrity of financial institutions. Financial fraud refers to any deliberate act intended to deceive for financial gain, while money laundering involves concealing the origins of illicit funds to make them appear legitimate. In recent years, the rapid digitization of financial services has increased the complexity and volume of transactions, creating new avenues for fraudulent activities. Transactions now happen at high volumes and speeds, often across borders and currencies, making it challenging for traditional monitoring systems to keep up.

Due to the prevalence of these crimes, financial institutions, governments, and regulatory bodies have prioritized fraud detection and prevention strategies. Despite substantial investment in anti-fraud technologies, conventional methods, such as rule-based systems, often fall short. Fraudsters continuously adapt their methods, and static rules are ill-suited to capture sophisticated, evolving fraud tactics. This gap highlights the need for advanced data-driven techniques that can dynamically detect fraud within complex transaction patterns.

1.2 Challenges in Fraud Detection within Financial Transactions

Detecting fraudulent behavior in financial transactions presents multiple challenges.

1. First, fraud detection models must operate in highly imbalanced datasets, as the proportion of fraudulent transactions is typically minute compared to legitimate ones. This imbalance often causes models to favor majority classes, leading to a higher rate of false negatives, where fraudulent transactions go undetected.

2. Second, financial transactions are frequently interconnected, with money flows and relationships linking various accounts. These connections can form intricate patterns, which can be challenging to analyze and interpret using traditional machine-learning techniques that only consider isolated data points.

3. Financial transactions occur in a time-sensitive manner, and the timing and sequence of events can provide critical clues for identifying fraudulent behavior. However, many traditional models struggle to effectively incorporate temporal information — such as the exact timestamp of a transaction or the duration between related events — into their learning process. Ignoring such time-based dependencies can lead to missed patterns, especially in cases where fraud unfolds rapidly or within specific time intervals.

4. Another core challenge is adaptability. Fraudsters continuously modify their tactics to circumvent detection systems, often exploiting weak points in the detection process. As a result, any effective detection system must be able to learn from evolving patterns and adapt to new fraud techniques.

Additionally, the need for real-time or near-real-time detection adds a layer of complexity, as large transaction datasets must be processed efficiently to identify anomalies without disrupting the flow of legitimate transactions.

1.3 Fraud Techniques Using Multiple Accounts

1. **Multiple Accounts for Smurfing (Money Laundering)**
   Smurfing, also known as ”structuring,” is a money laundering technique fraudsters use to avoid detection when transferring large sums of illicit money. Rather than moving a large amount in a single transaction, which may raise red flags in anti-money laundering (AML) systems, fraudsters break the transaction into smaller, more manageable amounts. This helps in evading attention from financial institutions and law enforcement agencies that monitor large or unusual transfers. Fraudsters set up multiple accounts either under their name or using fake identities. They then deposit smaller amounts of money into each of these accounts, carefully keeping the amounts below the threshold that would trigger AML systems or suspicion. After distributing the illicit funds across several accounts, the fraudster begins moving the money between these accounts, and eventually, they may transfer the funds to other individuals or offshore accounts. The complexity of the transactions, involving several accounts and intermediaries, makes it harder to trace the origin of the funds. The primary goal of smurfing is to launder illicit funds without attracting attention. By spreading the money across multiple accounts, the fraudster ensures that no single transaction is large enough to trigger a suspicious activity report. This

helps them ”clean” the money, eventually making it appear legitimate and difficult to trace back to its criminal origin.

2. **Complex Transfers**
   Complex transfers involve moving money through a series of transactions across multiple accounts to confuse investigators and obscure the source and destination of illicit funds. Fraudsters use this method to hide the flow of money, often leveraging multiple financial institutions, payment systems, and even international accounts. The complexity of the transfers creates a difficult-to-follow trail, making it harder for banks and law enforcement to identify fraudulent activity. A fraudster typically begins by transferring funds into an account they control. From there, they move the money through several intermediary accounts, which may be set up under fake identities or stolen credentials. These intermediary accounts can be spread across various banks or payment systems, both domestic and international. The fraudster might also use remittance services or cryptocurrencies to further obfuscate the transactions. By routing the funds through multiple layers, each transaction appears smaller or unrelated, complicating the investigation and making it more difficult to trace the money back to the fraudster. The objective behind using complex transfers is to create a confusing financial web that makes it difficult for investigators to trace the origin of illicit funds. This method also allows fraudsters to avoid detection by anti-fraud systems that monitor for large or suspicious transactions. By making the transactions appear normal and unconnected, the fraudster can move money freely while reducing the risk of being caught.

3. **Identity Theft and Account Takeovers**
   Fraudsters often use identity theft or account takeovers to gain control of multiple bank accounts, which allows them to conduct fraudulent activities without directly implicating themselves. By stealing personal information or hacking into a victim’s account, fraudsters can open accounts in the victim’s name or take control of their existing accounts to make unauthorized transactions. In the case of identity theft, fraudsters steal personal information such as Social Security numbers, credit card details, or bank account information. Using this data, they can open multiple accounts in the victim’s name or use the information to conduct fraudulent transactions. In account takeovers, fraudsters hack into existing bank accounts, often through phishing or social engineering, to change account credentials. Once in control, they can transfer funds, make purchases, or withdraw money from these accounts. Fraudsters may also use multiple accounts for further transactions, spreading the illicit money across various places to avoid detection. The goal of identity theft and account takeovers is to exploit stolen personal information or hacked accounts to conduct fraudulent activities. By using multiple accounts, fraudsters can cover their tracks, making it more difficult for investigators to trace the funds back to the original source. This technique enables fraudsters to siphon off funds without directly connecting the crime to their identity.

4. **Phishing Scams**
   Phishing scams involve fraudsters tricking individuals into revealing sensitive personal information such as usernames, passwords, and banking details. Through these scams, fraudsters gain access to victims' bank accounts and other financial services. Using this stolen information, they can then access multiple accounts to conduct fraudulent transactions, including transferring money or making unauthorized purchases. Fraudsters carry out phishing attacks by sending emails, text messages, or creating fake websites that mimic legitimate institutions like banks or online payment platforms. These communications often contain a sense of urgency, prompting victims to click on links or download attachments that lead to fraudulent websites. Once a victim enters their login credentials or personal information, fraudsters gain access to their bank account. The fraudster can then use the account to make unauthorized transactions or transfer funds to other accounts they control. Multiple accounts may be used to transfer the stolen funds, making it harder to trace the transactions. The goal of phishing scams is to deceive individuals into revealing their personal and banking information, which fraudsters can then use to gain unauthorized access to their accounts. By moving money between multiple accounts, fraudsters aim to obfuscate the trail and make it difficult for authorities to trace the fraudulent transactions back to the original source. This technique allows fraudsters to steal funds without being directly connected to the crime.

1.4 Motivation and Objectives of the Project

This project is motivated by the following challenges and seeks to utilize graph-based deep learning methods to improve fraud detection capabilities. By modeling financial transactions as a graph, where nodes represent accounts and edges signify transactional links, the project aims to capture relationships within the transaction network to reveal hidden patterns and suspicious behavior.

This project primarily explores the use of Graph Attention Networks (GATs) to detect fraudulent nodes (accounts) within a banking network. GATs offer several advantages over traditional Graph Convolutional Networks (GCNs) as they can dynamically assign different importance weights to different neighbors, allowing for more nuanced information aggregation. This attention mechanism is particularly valuable in fraud detection, where some transaction relationships may be more indicative of fraudulent behavior than others. GATs are especially effective in this context because they can:

1. Learn to focus on the most relevant neighboring nodes
2. Capture asymmetric relationships in the transaction network
3. Adapt to varying node neighborhoods without being constrained by fixed weights
4. Handle nodes with varying degrees of connectivity more effectively

1.5 Project Contribution and Research Questions

This project contributes a novel approach to fraud detection by integrating graph-based techniques with deep learning, specifically Graph Attention Networks (GATs), to leverage relationships within transaction data. The research addresses several key questions:

- **How can the structural properties of transaction networks be used to improve fraud detection?** By constructing a graph where nodes represent accounts and edges represent transactions, this project explores how the network topology and account interactions reveal fraud patterns.

- **To what extent can GATs improve the accuracy and robustness of fraud detection compared to traditional methods?** The ability of GATs to dynamically weight and aggregate features from neighboring nodes enables them to capture complex dependencies and selectively focus on relevant connections, potentially providing better fraud detection performance than methods that treat all relationships equally.

- **What features and relationships are most indicative of fraudulent activity in a transaction network?** By examining the attention weights assigned to different node attributes (for example, location, transaction count) and edge characteristics (e.g., amount, transaction type), this project investigates which relationships are most relevant for identifying fraud and how their importance varies across different contexts.

Chapter 2

Literature Review

This chapter provides a comprehensive review of the existing literature on Graph Neural Networks (GNNs) and their applications in fraud detection, with a particular focus on financial transactions. The review examines the evolution of graph-based approaches, the development of various GNN architectures, and their effectiveness in identifying fraudulent patterns in complex transaction networks.

2.1 Graphs

Graphs are a fundamental data structure used to model relationships between entities. In the context of fraud detection, transactions, users, and financial institutions can be represented as nodes, while the interactions or transactions between them form the edges. This graph-based representation allows for the exploration of structural and relational patterns that are often indicative of fraudulent behavior.

2.1.1 Types of Graph

Graphs can be broadly categorized into two types based on the nature of their nodes and edges: Homogeneous graphs and Heterogeneous graphs.

1. **Homogeneous Graphs**
   A homogeneous graph is a graph where all nodes and all edges are of the same type. This kind of graph is simpler and is typically used in scenarios with uniform entities and interactions. For example, a network where each node represents a bank account and each edge represents a financial transaction between accounts can be modeled as a homogeneous graph.

   Despite their simplicity, homogeneous graphs can capture complex structures and are widely used in early GNN models. However, they may fall short in representing the diversity of entities and relationships in real-world fraud detection tasks.

2. **Heterogeneous Graphs**

   A heterogeneous graph (also known as a heterogeneous information network) contains multiple types of nodes and/or multiple types of edges. In fraud detection, for example, the graph may contain nodes of types such as user, transaction, device, and merchant, with edges representing different kinds of interactions, like initiates, pays, logs in from, etc.

   Heterogeneous graphs are more expressive and allow for a richer representation of real-world systems, making them highly suitable for fraud detection tasks. They can model complex interdependencies between different types of entities, which is crucial for identifying fraudulent patterns that span across different domains.

2.2 Evolution of Graph Neural Networks

2.2.1 Foundational Work in Graph Neural Networks

The Graph Neural Network model, as introduced by Scarselli et al. [8], established the foundational framework for processing graph-structured data using neural networks. Their seminal work in 2009 proposed a recursive neural network architecture that could directly process graphs by iteratively updating node states until reaching a stable fixed point. This approach enabled the model to capture both node features and the structural information of the graph, addressing the limitations of traditional neural networks in handling non-Euclidean data structures.

The authors formalized the concept of information propagation across graph structures, where each node's representation is updated based on its neighbors' features and the graph topology. This recursive process allows the model to learn node representations that encode both local and global graph properties. The GNN model demonstrated effectiveness across various tasks, including node classification, graph classification, and link prediction, establishing a foundation for subsequent research in graph-based deep learning.

2.2.2 Theoretical Foundation of GNNs

Graph Neural Networks (GNNs) rely on two primary components for their operations: message passing (aggregation) and updating node states. Mathematically, this can be described as follows:

Message Passing (Aggregation Step)

Each node v aggregates information from its neighbors $N(v)$ at each layer l of the GNN. This process is generally represented as:

$$m_v^{(l)} = \text{AGGREGATE}^{(l)}\left(\{h_u^{(l-1)} : u \in N(v)\}\right)$$

Here, $h_u^{(l-1)}$ is the feature vector of a neighboring node u at the previous layer, and $m_v^{(l)}$ represents the aggregated message for node v. The aggregation function can be a simple sum, mean, or a more sophisticated attention-based operation.

Node Update (Transformation Step)

Once the node aggregates information from its neighbors, the aggregated message is combined with the node's previous state to update its feature representation:

$$h_v^{(l)} = \text{UPDATE}^{(l)}\left(h_v^{(l-1)}, m_v^{(l)}\right)$$

This step allows each node to retain its previous information while incorporating new information from its neighbors. The UPDATE function can involve a neural network layer, such as a fully connected layer or non-linear transformation (e.g., a ReLU activation), to improve expressiveness.

After several iterations (or layers), each node's feature vector encodes both its attributes and the structural information of its neighborhood, capturing dependencies and relationships across the graph.

2.2.3 Comprehensive Reviews of GNN Methods

As the field evolved, Zhou et al. [4] provided a comprehensive review of GNN methods and applications, categorizing the growing body of research into distinct frameworks and highlighting their theoretical foundations. Their review traced the development from early spectral approaches to more recent spatial-convolutional methods, offering insights into the strengths and limitations of different architectures.

The authors identified four primary categories of GNN models: recurrent graph neural networks (RecGNNs), convolutional graph neural networks (ConvGNNs), graph autoencoders (GAEs), and spatial-temporal graph neural networks (STGNNs). This categorization helped establish a unified understanding of the diverse approaches in the field and their respective applications across domains such as computer vision, natural language processing, and network analysis.

More recently, Khemani et al. [6] expanded on this foundation with an updated review that incorporated the latest advancements in GNN architectures, techniques, and applications. Their 2024 work emphasized the growing importance of GNNs in addressing complex real-world problems, including fraud detection in financial networks. The authors highlighted how GNNs have evolved to handle increasingly complex graph structures, including heterogeneous and dynamic graphs, which are particularly relevant for modeling financial transaction networks.

2.3 Specialized GNN Architectures

2.3.1 Graph Attention Networks

A significant advancement in GNN architecture came with the introduction of Graph Attention Networks (GATs) by Veličković et al. [10]. Unlike previous GNN models that treated all neighboring nodes equally during aggregation, GATs introduced an attention mechanism that assigns different importance weights to different neighbors. This innovation allowed the model to focus on the most relevant connections in the graph, significantly improving performance on various graph-based tasks.

The GAT architecture employs a self-attention mechanism over node features to compute attention coefficients, which are then used to weight the contributions of neighboring nodes during feature aggregation. The authors further enhanced the model's expressiveness by implementing multi-head attention, where multiple independent attention mechanisms are computed in parallel and their outputs concatenated or averaged. This approach stabilizes the learning process and enables the model to jointly attend to different aspects of the input graph.

Experimental results demonstrated that GATs outperformed previous state-of-the-art methods on benchmark datasets for node classification tasks, establishing attention mechanisms as a crucial component in modern GNN architectures. The ability to dynamically assign importance to different connections makes GATs particularly well-suited for fraud detection, where certain transaction relationships may be more indicative of fraudulent behavior than others.

2.3.2 GraphSAGE: Inductive Representation Learning

Hamilton et al. [3] introduced GraphSAGE (Graph SAmple and aggreGatE), an inductive framework for learning node embeddings in large graphs. Unlike transductive approaches that require all nodes to be present during training, GraphSAGE enables the generation of embeddings for previously unseen nodes, making it particularly valuable for dynamic graphs where new nodes appear over time—a common scenario in financial transaction networks.

The key innovation of GraphSAGE lies in its neighborhood sampling and aggregation strategy. Instead of using the entire neighborhood of a node, which can be computationally prohibitive in large graphs, GraphSAGE samples a fixed number of neighbors and applies learnable aggregation functions to combine their features. This approach significantly improves scalability while maintaining representational power.

The authors proposed several aggregation functions, including mean aggregation, LSTM-based aggregation, and pooling-based aggregation, each offering different trade-offs between computational efficiency and expressive power. Experimental results demonstrated that GraphSAGE could effectively generate embeddings for unseen nodes across various domains, including citation networks and Reddit posts.

For financial fraud detection, GraphSAGE's inductive capabilities are particularly valuable, as they enable the model to generalize to new accounts and transactions without requiring retraining. This adaptability is crucial in real-world fraud detection systems, where new entities continuously enter the network and fraudulent patterns evolve over time.

2.3.3 Graph Attention Networks

Building upon the success of GATs, Chen and Chen [5] proposed the Edge-Featured Graph Attention Network (EGAT), which extends the attention mechanism to incorporate edge features. In many real-world graphs, edges contain valuable information that can enhance model performance. For financial transaction networks, edge features such as transaction amount, timestamp, and payment type provide critical context for identifying suspicious patterns.

The EGAT model modifies the attention mechanism to consider both node and edge features when computing attention coefficients. This allows the model to capture more nuanced relationships between nodes, taking into account the characteristics of the connections themselves. The authors demonstrated that incorporating edge features significantly improved performance on various graph-based tasks, particularly in scenarios where edge attributes contain relevant information for the task at hand.

This edge-enhanced approach is particularly valuable for fraud detection in financial networks, where transaction details (represented as edge features) often contain strong signals of fraudulent activity. By integrating these features into the attention mechanism, the model can better identify suspicious transaction patterns and improve detection accuracy.

2.3.4 Heterogeneous Graph Neural Networks

As graph-based applications expanded to more complex domains, the need for models capable of handling heterogeneous graphs—where nodes and edges can be of different types—became increasingly apparent. Zhang et al. [11] addressed this challenge with the Heterogeneous Graph Neural Network (HetGNN), which leverages both structure and content information to learn representations for heterogeneous graphs.

HetGNN employs a two-stage process: first, it samples heterogeneous neighbors for each node using random walks with restart, capturing both homogeneous and heterogeneous relationships; second, it uses a neural network architecture with type-specific aggregators to encode node content and structural information. This approach enables the model to preserve both the heterogeneity of the graph and the semantic information associated with different node types.

Building on this foundation, Hu et al. [13] introduced the Heterogeneous Graph Transformer (HGT), which adapts the transformer architecture for heterogeneous graph learning. The HGT model introduces type-specific parameters for different node and edge types, allowing it to capture the unique characteristics of each entity and relationship in the graph. The model employs a hierarchical attention mechanism that considers node type, edge type, and target node type when computing attention scores, enabling it to learn different patterns for different types of interactions.

These heterogeneous approaches are particularly relevant for financial fraud detection, where transaction networks often involve multiple types of entities (e.g., individual accounts, business accounts, merchants) and relationships (e.g., deposits, transfers, purchases). By modeling these heterogeneous elements explicitly, these models can capture more complex fraud patterns that may span different types of accounts and transactions.

2.4 Edge Features in Graph Neural Networks

2.4.1 Exploiting Edge Features

Gong and Cheng [7] further explored the importance of edge features in GNNs, proposing methods to effectively incorporate edge information into the message-passing framework. Their work demonstrated that edge features can significantly enhance model performance across various tasks, particularly when the relationships between nodes contain relevant information for the task at hand.

The authors introduced several approaches for integrating edge features, including edge feature transformation, edge-node feature fusion, and edge-conditioned convolution. These methods enable the model to consider both the structure of the graph and the characteristics of the connections when learning node representations, resulting in more expressive and informative embeddings.

For financial fraud detection, these edge-centric approaches offer a powerful framework for modeling transaction networks, where the details of each transaction (amount, time, method) provide critical signals for identifying suspicious patterns. By explicitly incorporating these edge features into the learning process, the model can better distinguish between legitimate and fraudulent transaction patterns.

2.5 Addressing Imbalance in Graph Datasets

2.5.1 Challenges of Imbalanced Data

A significant challenge in fraud detection is the inherent imbalance in the data, where fraudulent transactions typically represent a small minority of the overall dataset. Fofanah et al. [1] addressed this issue by introducing GATE-GNN, a model specifically designed to handle imbalanced graph datasets through graph ensemble weight attention and transfer learning.

The authors proposed a two-stage approach: first, training multiple GNN models on balanced subsets of the data, and then combining their predictions using an attention mechanism that assigns different weights to different models based on their performance on the minority class. This ensemble approach helps mitigate the bias towards the majority class that often occurs in imbalanced datasets.

Additionally, the authors explored transfer learning techniques to leverage knowledge from related domains or tasks, further improving performance on the minority class. By pre-training the model on a larger, balanced dataset and then fine-tuning it on the target imbalanced dataset, the model can learn more robust representations that generalize better to the minority class.

These approaches are particularly valuable for financial fraud detection, where the rarity of fraudulent transactions can make them difficult to detect using standard machine learning techniques. By explicitly addressing the class imbalance issue, GATE-GNN and similar methods can improve the detection of fraudulent patterns while maintaining reasonable false positive rates.

2.6 Applications of GNNs in Fraud Detection

2.6.1 GNNs for Financial Fraud Detection

The application of GNNs to financial fraud detection has gained significant attention in recent years, with several studies demonstrating their effectiveness in identifying suspicious transaction patterns. Zhou et al. [12] reviewed various applications of GNNs across domains, highlighting their potential for fraud detection in financial networks.

The authors noted that GNNs are particularly well-suited for fraud detection due to their ability to capture the relational nature of fraudulent activities. Financial fraud often involves complex networks of transactions designed to obscure the flow of funds, making graph-based approaches ideal for uncovering these hidden patterns. By modeling accounts as nodes and transactions as edges, GNNs can learn to identify suspicious subgraphs within the larger transaction network.

Several key advantages of GNNs for fraud detection were identified:

- **Relational Inductive Bias**: GNNs naturally incorporate the structure of the transaction network into the learning process, enabling them to capture patterns that span multiple accounts and transactions.
- **Feature Integration**: GNNs can effectively combine node features (account characteristics) and edge features (transaction details) to learn more comprehensive representations.
- **Scalability**: Recent advances in GNN architectures, such as GraphSAGE [3] and cluster-GCN, enable efficient processing of large-scale transaction networks, making them practical for real-world applications.

2.7 Graph Attention Networks (GATs) in Fraud Detection

For this project, Graph Attention Networks (GATs) are used as the primary GNN architecture. GATs excel at tasks like node classification and link prediction, which are directly applicable to fraud detection in banking transaction networks. Unlike traditional GCNs, GATs introduce an attention mechanism that allows the model to assign different importance weights to different neighboring nodes, making them particularly effective at identifying suspicious patterns in transaction networks where some connections are more indicative of fraud than others.

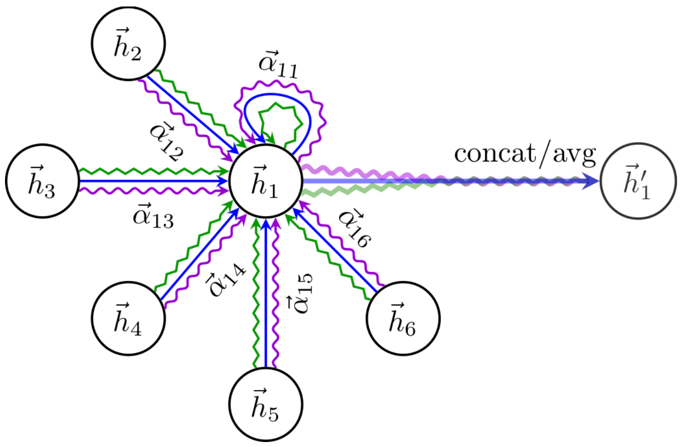


Figure 2.1: Graph Attention Networks: An illustration of multi-head attention (with K = 3 heads)[10]

The architecture of GATs involves multiple graph attention layers, where each layer computes attention coefficients to weigh the importance of neighboring nodes during feature aggregation. This attention-based approach allows the model to:

- Dynamically focus on the most relevant transaction relationships
- Capture asymmetric importance in node relationships
- Learn complex patterns without being constrained by fixed neighborhood weightings

This multi-layer architecture allows the model to capture both immediate and multi-hop relationships among accounts, with the attention mechanism helping to identify the most suspicious trails indicative of laundering or fraudulent activity. The resulting node embeddings, enriched with attention-weighted information from the neighborhood, are used to classify nodes (accounts) as either fraudulent or legitimate.

2.7.1 Spatial-Temporal Attention for Fraud Detection

A critical dimension of financial fraud detection is the temporal aspect of transactions, as fraudulent activities often exhibit distinctive temporal patterns. Cheng et al. [2] addressed this challenge by introducing a Spatial-Temporal Attention Network (STAN) specifically designed for fraud detection in financial transactions.

The STAN model combines graph neural networks with attention mechanisms to capture both spatial (structural) and temporal dependencies in transaction data. The spatial attention component identifies important connections in the transaction graph, while the temporal attention component focuses on relevant historical patterns. This dual-attention approach enables the model to detect complex fraud patterns that manifest across both space and time.

The authors evaluated their model on real-world credit card transaction data, demonstrating significant improvements over traditional methods and standard GNN approaches. The results highlighted the importance of jointly modeling spatial and temporal information for effective fraud detection, as many fraudulent patterns involve specific sequences of transactions across multiple accounts.

This spatial-temporal approach is particularly valuable for detecting sophisticated fraud schemes, such as money laundering or organized fraud rings, where the temporal ordering of transactions across multiple accounts is a key indicator of suspicious activity. By explicitly modeling these temporal dependencies, STAN and similar models can identify patterns that would be missed by purely structural approaches.

2.8 Transformer-based Approaches in Graph Learning

2.8.1 Transformers and Their Impact on GNNs

The Transformer architecture, introduced by Vaswani et al. [9], has revolutionized natural language processing and subsequently influenced various domains, including graph representation learning. Originally designed for sequence modeling tasks, Transformers rely on self-attention mechanisms to capture dependencies between elements in a sequence, regardless of their distance from each other.

This ability to model long-range dependencies makes Transformers particularly attractive for graph learning, where capturing interactions between distant nodes can be crucial for certain tasks. Several recent works have adapted the Transformer architecture for graph-structured data, combining the strengths of attention-based models with the structural inductive biases of GNNs.

The key innovation of Transformers—the self-attention mechanism—aligns naturally with the goals of GNNs, as both aim to selectively aggregate information from related entities. However, while traditional GNNs typically focus on local neighborhoods, Transformer-based approaches can capture global dependencies across the entire graph, potentially addressing the limited receptive field issue that affects many GNN architectures.

For fraud detection, Transformer-based graph models offer several advantages, including the ability to capture complex patterns across distant parts of the transaction network and to model higher-order interactions between multiple entities. These capabilities are particularly valuable for detecting sophisticated fraud schemes that involve coordinated activities across seemingly unrelated accounts

2.9 Research Gaps and Future Directions

Despite the significant advances in applying GNNs to fraud detection, several research gaps remain to be addressed:

- **Explainability**: While GNNs demonstrate strong performance in fraud detection, their decision-making process often lacks transparency. Developing methods to explain GNN predictions is crucial for practical applications in financial fraud detection, where understanding why a transaction was flagged as suspicious is essential for investigation and regulatory compliance.
- **Adaptive Learning**: Fraudulent patterns evolve over time as perpetrators adapt to detection systems. Developing GNN models that can continuously learn from new data and adapt to emerging fraud patterns remains an important research direction.
- **Multi-modal Data Integration**: Financial transactions often come with additional data modalities, such as text descriptions or customer interactions. Integrating these diverse data sources into GNN-based fraud detection systems presents both challenges and opportunities for improving detection accuracy.
- **Privacy-Preserving Learning**: Financial data is highly sensitive, and privacy concerns can limit the sharing and use of transaction data for fraud detection. Developing privacy-preserving GNN methods that can learn from encrypted or federated data could address these concerns while maintaining detection effectiveness.
- **Scalability for Real-time Detection**: While recent advances like GraphSAGE have improved the scalability of GNNs, applying these models for real-time fraud detection in large-scale transaction networks remains challenging. Developing more efficient inference methods and incremental learning approaches could address this gap.

2.10 Summary

The literature review reveals a rich landscape of research on Graph Neural Networks and their applications to fraud detection. From the foundational work on GNN models to specialized architectures like GATs and GraphSAGE, significant progress has been made in developing methods capable of capturing the complex patterns indicative of fraudulent activities in financial networks.

The integration of attention mechanisms, edge features, and temporal information has further enhanced the effectiveness of GNNs for fraud detection, enabling more nuanced modeling of transaction networks. Additionally, approaches for addressing class imbalance and handling heterogeneous graphs have made these methods more applicable to real-world fraud detection scenarios.

The emergence of Transformer-based approaches has opened new avenues for graph representation learning, offering potential solutions to some of the limitations of traditional GNNs. By combining the global modeling capabilities of Transformers with the structural inductive biases of GNNs, these hybrid approaches show promise for improving fraud detection performance.

Despite these advances, challenges remain in areas such as explainability, adaptive learning, privacy preservation, and scalability. Addressing these challenges represents promising directions for future research, with the potential to further improve the effectiveness and practicality of GNN-based fraud detection systems.

This literature review provides the foundation for the current research, which builds upon these existing approaches to develop an enhanced Graph Attention Network model specifically tailored for banking fraud detection. By incorporating edge features, addressing class imbalance, and leveraging the spatial-temporal properties of transaction networks, this research aims to advance the state-of-the-art in financial fraud detection using graph-based approaches.

Chapter 3

Methodology

The methodology of this project involves several key steps, from data preprocessing to model training and evaluation. This section outlines the process used to convert transactional data into a graph, engineer features, and apply Graph Attention Networks (GATs) to detect fraudulent activity.

3.1 Dataset Description

The dataset utilized in this project serves as the foundation for constructing a graph representation of financial transactions, with each record detailing account interactions in a transactional network. This dataset includes multiple key features that provide insights into transactional behavior, enabling the application of Graph Convolutional Networks (GCNs) to detect fraudulent accounts. The following sections outline the dataset's features, pre-processing steps, and challenges associated with real-world transaction data.

3.1.1 Description of Dataset Features

The dataset contains the following columns:

1. **Time**: Records the transaction timestamp, capturing the exact time of day when the transaction occurred. This is particularly useful for establishing a trial's temporal sequence of transactions.

2. **Date**: Specifies the transaction date, later decomposed into multiple columns for effective feature extraction. The date enables chronological ordering in the transaction network.

3. **Sender_account**: The unique identifier for the account initiating the transaction. This feature is crucial in defining the source node in the graph representation.

4. **Receiver_account**: The unique identifier for the account receiving the transaction. This feature defines the target node in the graph and, combined with the sender account, establishes the directed edges.

5. **Amount**: The monetary value of the transaction. As an edge feature in the graph, the amount indicates the volume of funds transferred between nodes, which can signal suspiciously large or irregular transactions.

6. **Payment_currency** and **Received_currency**: These columns denote the currency used for payment and receipt in the transaction, respectively. Currency information helps highlight cross-border or multi-currency transactions, which could indicate money laundering.

7. **Sender_bank_location** and **Receiver_bank_location**: These features identify the locations of the sender and receiver banks. Geographical location data helps in detecting suspicious flows of money across high-risk areas or tax havens.

8. **Payment_type**: Specifies the transaction method (e.g., Cash Deposit, Online Transfer), offering insights into transaction patterns that may vary depending on fraud typology. Different payment types could correlate with distinct fraud schemes.

9. **Is_laundering**: A binary indicator where '1' denotes a suspicious transaction linked to money laundering, and '0' represents a normal transaction. This serves as the target label for the node classification model.

10. **Laundering_type**: Describes the type of laundering activity, providing additional information for classification and aiding in model interpretability by indicating different fraud typologies.

These features enable the construction of a detailed graph representation where nodes correspond to accounts, and directed edges signify transactions. The features serve either as node attributes (e.g., account-specific attributes like location) or edge features (e.g., transaction-specific attributes like amount and date).

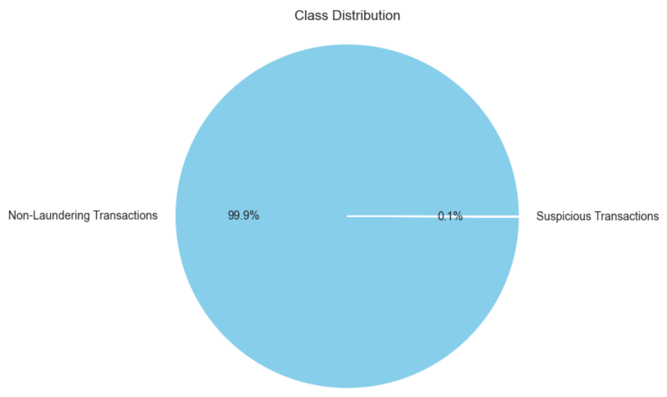
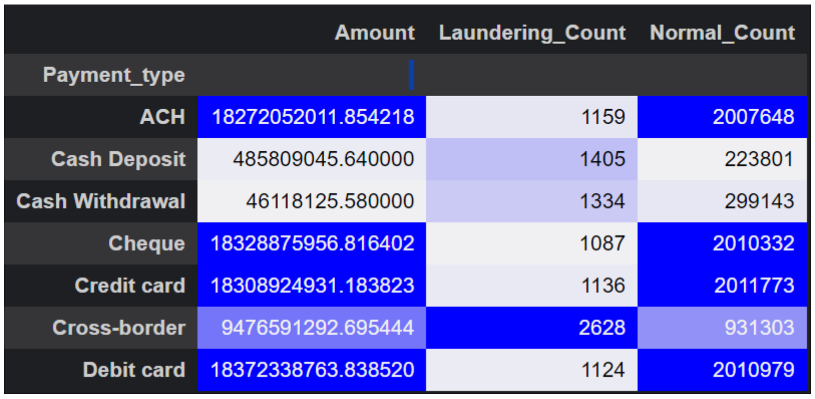


Figure 3.1: Dataset Distribution

The dataset consisted of 95,04,852 transactions with the above-mentioned 12 features. From these transactions, the following is the distribution of laundering and non-laundering transactions.

- 9494979 - Non-laundering(99.9%).
- 9873 - Laundering(0.1%).



	Amount	Laundering_Count	Normal_Count
Payment_type			
ACH	18272052011.854218	1159	2007648
Cash Deposit	485809045.640000	1405	223801
Cash Withdrawal	46118125.580000	1334	299143
Cheque	18328875956.816402	1087	2010332
Credit card	18308924931.183823	1136	2011773
Cross-border	9476591292.695444	2628	931303
Debit card	18372338763.838520	1124	2010979

Figure 3.2: Distribution of fraud and no-fraud transactions according to payment type

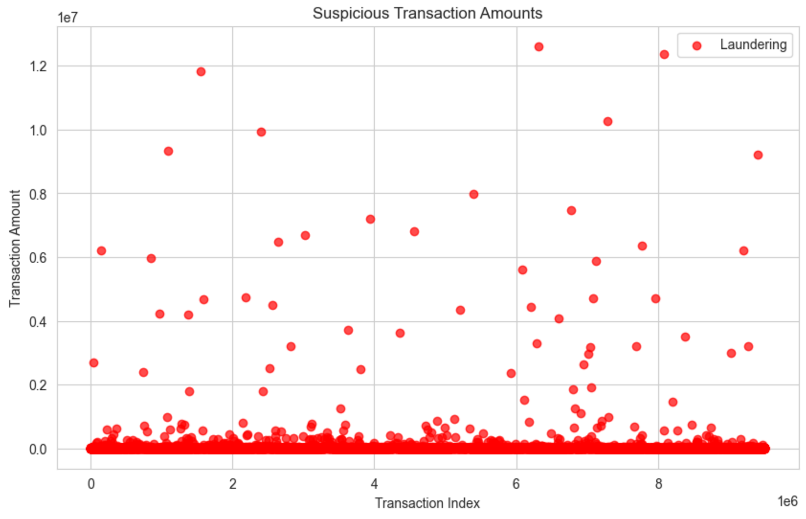


Figure 3.3: Suspicious Transaction Amounts

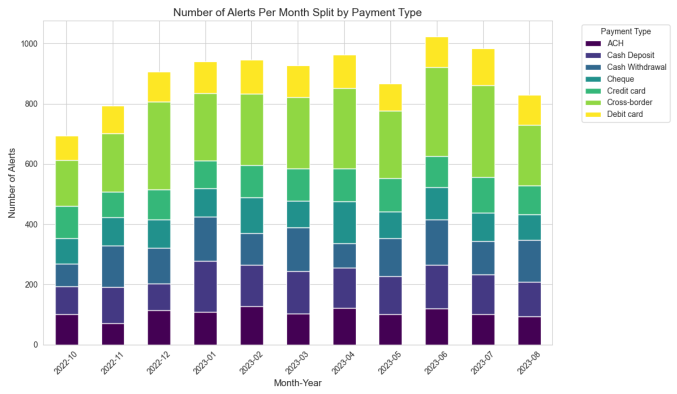


Figure 3.4: Number of Alerts Per Month Split by Payment Type

3.1.2 Notable Pre-processing Steps and Assumptions

To facilitate efficient model training and optimize feature relevance, several pre-processing steps were undertaken. Below are the key steps:

1. **Date Conversion and Expansion**:

   The Date column was converted to a date-time format to enable temporal analysis. From this, additional columns were extracted:

   - **Year**: Represents the year of the transaction.
   - **Month**: Indicates the month of the transaction, capturing seasonal patterns.
   - **Day**: Specifies the day, useful for detecting weekday versus weekend transaction patterns.

2. **Encoding Categorical Features**:

   To allow the model to process categorical features, columns like `Payment_type`, `Sender_bank_location`, and `Receiver_bank_location` were encoded using Label Encoding.

3. **Scaling of Continuous Features**: Continuous features, such as `Amount`, were scaled to bring them into a consistent range, which is essential for models like GCNs that are sensitive to feature magnitudes. Scaling helps in ensuring that high-value transactions do not disproportionately impact model predictions.

4. **Constructing Edge and Node Features**: Transaction amount and date were used as edge features, while sender and receiver location, and transaction degree (in-degree and out-degree), were defined as node features. This distinction in feature assignment is crucial for accurately modeling relationships and patterns within the transaction graph.

3.2 Creation of Subset of the Original Dataset

To develop an effective dataset for fraud detection, a subset of the original data was created to address the significant class imbalance in transaction labels. The original dataset contains over **9.4 million transaction records**, classified as follows:

- **Fraud transactions**: 9,873 samples
- **Non-fraud transactions**: 9,494,979 samples

Given the goal of ensuring meaningful representation for accounts involved in fraudulent activity, the non-fraud samples were split into two distinct parts:

1. **First Part**: This subset includes non-fraud transactions where accounts (either as sender or receiver) were also associated with fraudulent transactions. By including these, we establish connections within the data conducive to graph-based analysis, as these accounts represent cases of potential indirect fraud involvement.
2. **Second Part**: This subset contains non-fraud transactions from accounts with no involvement in fraudulent transactions, thus serving as a baseline for typical non-fraud behavior.

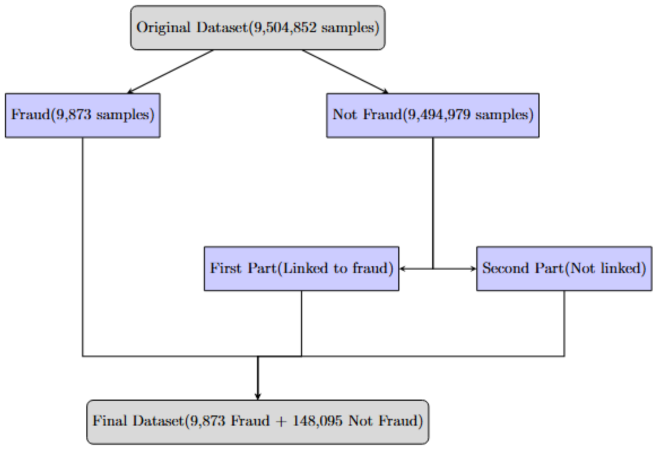

Original Dataset(9,504,852 samples)
Fraud(9,873 samples)
Not Fraud(9,494,979 samples)
First Part(Linked to fraud)
Second Part(Not linked)
Final Dataset(9,873 Fraud + 148,095 Not Fraud)

Figure 3.5: Taking subset of Original Dataset

For constructing the final dataset, a subset of samples was chosen to ensure both class representation and manageable dataset size:

- **All 9,873 fraud samples** were retained.
- **148,095 non-fraud samples** were selected, representing 15 times the number of fraud samples.

To achieve a representative non-fraud sample, a stratified selection approach was employed:

- **40%** of the non-fraud samples were sourced from the **First Part** (accounts associated with fraud),
- **60%** of the non-fraud samples were sourced from the **Second Part** (accounts not associated with fraud).

The dataset used for fraud detection was carefully constructed with a graph-based approach. The graph consists of nodes and edges, where nodes represent accounts involved in transactions, and edges represent the transactions between those accounts. The key characteristics of the graph are:

- **Number of nodes**: 138,388

  - Each node represents a unique account involved in a transaction (either as a sender or receiver).

- **Number of edges**: 157,968
  - Each edge represents a transaction between two accounts. The total number of edges matches the size of the dataset, as each sample (whether fraud or non-fraud) corresponds to a transaction between two accounts.

The construction of this graph follows the stratification approach used in the dataset creation. The nodes are connected based on transactions, with fraudulent and non-fraudulent accounts forming edges between them. The graph is used to capture the relationships between accounts involved in fraud or not, enabling more sophisticated graph-based fraud detection techniques.

3.3 Data Preprocessing

Data preprocessing is essential for ensuring that the dataset is clean, consistent, and ready for analysis. The steps taken during preprocessing include:

1. **Data Cleaning**: Missing values for key attributes, such as account location or transaction amount, were either imputed with default values or removed, depending on the relevance of the feature to fraud detection. For instance, default codes were used for missing location or currency data, ensuring that such entries could still be used in the graph.
2. **Feature Extraction**: To capture temporal information, the transaction date was converted into separate year, month, and day columns. Additionally, a continuous representation of the date was created to allow the model to interpret temporal relationships in the data.
3. **Normalization**: Continuous variables, such as transaction amounts, were normalized to improve model stability and performance. This helps the GAT model better handle variations in transaction amounts across different accounts.

3.4 Graph Construction

Graph construction is a critical step in leveraging Graph Neural Networks (GNNs) for fraud detection, as it defines how entities and their relationships are represented for modeling. In the context of financial transaction data, constructing a graph involves representing accounts *accounts*, *transactions*, and their interactions in a structured format suitable for GNNs. Two primary approaches are employed: **homogeneous graph construction** and **heterogeneous graph construction**. Each method offers distinct advantages in capturing the relational dynamics of transactional data, particularly for identifying fraudulent activities. This section details both approaches, their methodologies, and their relevance to fraud detection.

3.4.1 Homogeneous Graph Construction

A homogeneous graph represents all entities as a single type of node (e.g., accounts) and relationships as a single type of edge (e.g., transactions). This approach simplifies the graph structure, making it computationally efficient and suitable for scenarios where the focus is on interactions between accounts.

3.4.1.1 Defining Nodes and Edges

In this project, nodes represent either the sender or receiver accounts involved in the transactions and edges represent individual transactions. This node-edge relationship is defined as follows:

- **Nodes**: Each node corresponds to a unique account involved in one or more transactions. The nodes have several attributes (or features) that capture information about the account, such as location, currency type, transaction frequency, and suspicious behavior count.
- **Edges**: Each edge represents a single transaction from a sender account to a receiver account. Edge features include transaction details like the amount, payment type, and a numeric representation of the transaction date. These edges are directional, where an edge from node A to node B indicates a transaction from account A to account B.

3.4.1.2 Node and Edge Features

To enable effective learning, each node and edge in the graph is associated with specific features derived from the dataset. These features help represent the unique characteristics of accounts and transactions, which are crucial for identifying suspicious patterns.

- **Node Features**:
  - **Sender Location and Receiver Location**: The geographical locations of the sender and receiver accounts are encoded as numeric values. If a location is unavailable, it is represented by a default value (e.g., 18). If there are multiple locations used by the sender then the most frequently used location is taken as the final sender location.
  - **Payment Currency and Received Currency**: The currencies involved in the transactions are also encoded numerically. In cases where no currency information is available, a default value (e.g., 12) is used. If multiple currencies are being used by the sender then the most frequently used currency is taken as a feature.

  - **Out-degree and In-degree**: The out-degree of a node represents the number of transactions sent from the account, while the in-degree represents the number of transactions received by the account. These values are indicative of transaction frequency and can signal potential red flags if they fall outside typical ranges.

  These features are aggregated from transaction data, using statistical measures like the mode for categorical attributes (e.g., location, currency) and counts for degree-based metrics. For accounts with no transactions (isolated nodes), feature vectors are set to zero to ensure consistency.

- **Node Labels** Each node is assigned a binary label indicating involvement in fraudulent activity (1 for laundering, 0 otherwise). The label is determined by checking if the account participates in any transaction marked as fraudulent, ensuring that the graph captures the ground truth for supervised learning.

- **Edge Features**:

  - **Transaction Amount**: The transaction amount is represented as a continuous feature, capturing the scale of each transaction.
  - **Payment Type**: This categorical feature encodes the type of transaction, such as cash deposit, transfer, or online payment. This feature can help distinguish between typical and suspicious transaction types.
  - **Date Representation**: The date of each transaction is converted to a continuous value representing the year, month, and day as a numeric date format:

    $$\text{Year} + \frac{\text{Month}}{12} + \frac{\text{Day}}{365}$$

    This format helps capture the timing of transactions in a compact and computationally efficient way, enabling the model to recognize patterns over time.

3.4.2 Heterogeneous Graph Construction

A heterogeneous graph extends the homogeneous approach by modeling multiple entity types (e.g., accounts and transactions) and their relationships as distinct node and edge types. This method is particularly suited for complex financial networks where transactions and accounts have unique properties and interactions.

3.4.2.1 Node Types and Features

The heterogeneous graph includes two primary node types:

- **User Nodes**: Represent accounts, similar to the homogeneous graph. Each user node is associated with features derived from their transactional history:
  - **Currency**: The most frequent currency used in sending or receiving transactions, encoded as a one-hot vector to capture categorical diversity (e.g., 13 currency types).
  - **Location**: The most frequent bank location associated with the account, is also one-hot encoded (e.g., 18 location types).

  These features are computed by aggregating all transactions involving the account, with the most frequent currency and location selected to represent the account's dominant behavior. For efficiency, this process can leverage parallel processing to handle large datasets, where transaction data is split into chunks, processed independently, and then combined.

- **Transaction Nodes**: Represent individual transactions, each with its feature vector:
  - **Amount**: The transaction's monetary value, normalized or scaled as needed.
  - **Payment Type**: Encoded as a one-hot vector (e.g., 7 payment types) to distinguish transaction categories.
  - **Date**: Stored as a string (e.g., "YYYY-MM-DD") or numerical value for temporal analysis.
  - **Fraud Label**: A binary indicator (1 for laundering, 0 otherwise) assigned to each transaction, serving as the ground truth for fraud detection.

  Transaction nodes enable the graph to explicitly model each financial event, preserving detailed information that might be aggregated or lost in a homogeneous graph.

3.4.2.2 Edge Types and Connections

The heterogeneous graph defines two directed edge types to capture the relationships between users and transactions:

- **User-to-Transaction (Sends)**: Connects a user node (sender) to a transaction node, indicating that the account initiated the transaction. The edge index is constructed by mapping each transaction to the sender's user index and the transaction's index.

- **Transaction-to-User (Received By)**: Connects a transaction node to a user node (receiver), indicating that the account received the funds. Similarly, the edge index maps each transaction to the receiver's user index.

These edge types create a bipartite structure, where users and transactions form a network of interactions. No direct user-to-user edges exist, as all relationships are mediated through transaction nodes, preserving the granularity of the data.

3.4.3 Constructing Transaction Trails

Fraudulent behavior is often identified not from individual transactions but from patterns across multiple transactions. To capture this, transaction trails are created within the graph, where a sequence of transactions (edges) connects a series of accounts (nodes). For example, if Account A sends money to Account B, which in turn sends money to Account C, a transaction trail is formed as:

$$A \rightarrow B \rightarrow C \rightarrow ...$$

This enables the model to learn dependencies along transaction paths, helping identify money-laundering chains and other fraudulent trails. Longer transaction trails are particularly useful for detecting complex schemes, as they reveal intricate transaction relationships and account dependencies.

3.5 Model Selection

3.5.1 Graph Attention Networks (GATs)

Graph Attention Networks (GATs) were chosen for this project because of their effectiveness in learning from graph-structured data and their ability to assign different importance to different nodes in the neighborhood. Unlike traditional GCNs, which treat all neighbors equally, GATs apply attention mechanisms to weigh the contributions of different neighbors dynamically. Key components of the GAT architecture include:

- **Attention-based Message Passing**: Each node computes attention coefficients with its neighbors and uses these to weigh the importance of different nodes during feature aggregation. This enables the model to focus on the most relevant connections and identify suspicious patterns more effectively.

- **Multi-head Attention**: Multiple attention mechanisms run in parallel, allowing the model to capture different aspects of node relationships simultaneously and stabilize the learning process.

- **Node Update**: Nodes update their representations based on the attention-weighted aggregated information, allowing each node's representation to evolve through the network layers while preserving the most relevant neighborhood information.

- **Self-attention Mechanism**: Each node can attend to its features, helping preserve important local information while incorporating neighborhood context.

3.5.2 Spatial Temporal Aware Graph Transformer

The model architecture for fraud detection in financial transaction networks integrates Graph Neural Networks (GNNs), temporal encoding, and transformer-based mechanisms to effectively capture the complex relational and temporal patterns inherent in transaction data. This approach leverages a heterogeneous graph structure, where accounts (users) and transactions are distinct node types, connected by directed edges representing sending and receiving relationships. The model processes node features, edge connections, and temporal information to predict whether a transaction is fraudulent, outputting a binary classification (fraudulent or non-fraudulent). This section details the components of the model, their functionalities, and their contributions to fraud detection.

3.5.2.1 Temporal Encoding

Temporal information is critical in fraud detection, as fraudulent activities often exhibit distinct temporal patterns, such as rapid transaction sequences. The model incorporates a temporal encoding mechanism to embed transaction timestamps into a high-dimensional space, enabling the capture of temporal dynamics.

Encoding Mechanism The temporal encoding transforms scalar timestamps into dense vectors using a positional encoding scheme inspired by transformer models. For a given timestamp t, the encoding generates a d-dimensional vector, where d is the embedding dimension (e.g., 64). The encoding is computed as follows:

$$\mathrm{PE}(t, 2i) = \cos\left(\frac{t}{10000^{2i/d}}\right), \quad \mathrm{PE}(t, 2i+1) = \sin\left(\frac{t}{10000^{2i/d}}\right),$$

for $i = 0, 1, \ldots, \lfloor d/2 \rfloor$. The sine and cosine functions alternate across dimensions, with frequencies determined by the divisor $10000^{2i/d}$. This creates a unique, continuous representation of time that captures both short-term and long-term temporal relationships. The encoded vectors are passed through a linear layer to align their dimensionality with other embeddings in the model, ensuring compatibility with subsequent layers.

Integration with Transaction Features The temporal encoding is applied to the timestamps of transaction nodes, which are typically stored as dates (e.g., “YYYY-MM-DD”) and converted to numerical timestamps. The resulting temporal embeddings are added to the initial transaction node features, enriching them with temporal context. This allows the model to differentiate transactions occurring at different times, facilitating the detection of suspicious patterns, such as bursts of activity within a short timeframe.

3.5.2.2 Initial Feature Embedding

The model begins by transforming the raw features of user and transaction nodes into a unified embedding space, ensuring that heterogeneous node types are represented consistently for graph-based processing.

User Node Embedding User nodes, representing accounts, have features such as the most frequent currency and bank location, typically encoded as one-hot vectors (e.g., 13 currency types, 18 location types, yielding a 31-dimensional feature vector). These features are passed through a linear layer to project them into a hidden dimension (e.g., 64). This transformation preserves the categorical information while aligning the embeddings with the model's hidden space.

Transaction Node Embedding Transaction nodes have features including the transaction amount and payment type, often represented as a combination of numerical values and one-hot encoded categories (e.g., 8-dimensional features with 7 payment types). Similar to user nodes, these features are projected into the hidden dimension using a linear layer with ReLU activation. The temporal encoding is then added to the transaction embeddings, enhancing them with temporal information.

3.5.2.3 Graph Neural Network Layers

The core of the model is a multi-layer GNN designed to propagate and aggregate information across the heterogeneous graph, capturing relational patterns between users and transactions.

- **Heterogeneous Convolution** The GNN operates on the heterogeneous graph, which includes two edge types: **user-to-transaction** ("sends") and **transaction-to-user** ("received by"). Each edge type is associated with a relation-specific convolution, implemented using a graph convolutional operator (e.g., SAGEConv). For each layer, the convolution updates node embeddings as follows:

  $$h_v^{(l+1)} = W_r^{(l)} h_v^{(l)} + \sum_{u \in \mathcal{N}_r(v)} \frac{1}{|\mathcal{N}_r(v)|} W_r^{(l)} h_u^{(l)},$$

  where $h_v^{(l)}$ is the embedding of node v at layer l, R is the set of edge types, $\mathcal{N}_r(v)$ is the set of neighbors of v under relation r, and $W_r^{(l)}$ is a learnable weight matrix for relation r. The first term preserves the node's own embedding, while the second term aggregates information from neighboring nodes, weighted by the relation-specific transformation.

  The model employs multiple GNN layers (e.g., 2 layers), with each layer refining the node embeddings by incorporating higher-order neighborhood information. The heterogeneous convolution ensures that user and transaction nodes are updated differently based on their roles in the graph (sender or receiver).

- **Relation-Level Attention** To prioritize relevant relationships, the model introduces a relation-level attention mechanism within each GNN layer, focusing on transaction nodes as the primary entities for fraud classification. The attention mechanism computes scores for each edge type to determine their relative importance in updating transaction node embeddings.

  For each relation r, an attention score is calculated using the transaction node embeddings h_v post-convolution. The embeddings are transformed using a learnable linear layer W and bias b, followed by a tanh activation. A learnable query vector q is then used to compute the score:

  $$\text{score}_r = \frac{1}{N}\left(q^T \cdot \tanh(Wh + b)\right),$$

  where h is the matrix of transaction node embeddings (shape: $N \times d$, with N transaction nodes and d embedding dimensions), and the matrix multiplication $q^T \cdot \tanh(Wh + b)$ is averaged over the node dimension. The scores for all relations are stacked and normalized using a softmax function to obtain attention weights α_r:

  $$\alpha_r = \frac{\exp(\text{score}_r)}{\sum_{r' \in R} \exp(\text{score}_{r'})}.$$

  These weights are intended to combine relation-specific transaction embeddings, emphasizing relations more indicative of fraudulent behavior, such as high-value transactions initiated by specific users. However, in the implementation, the transaction embeddings post-convolution are currently identical across relations, and the attention weights modulate their contribution as:

  $$h_{\text{trans}}^{(l+1)} = \sum_{r \in R} \alpha_r \cdot \text{ReLU}(h_{\text{trans}}^{(l+1)}),$$

  where $h_{\text{trans}}^{(l+1)}$ is the transaction embedding after the heterogeneous convolution. This suggests that the attention mechanism effectively reweights the same embeddings, potentially limiting its ability to differentiate relation-specific contributions. Ideally, relation-specific embeddings should be used to fully leverage the heterogeneous graph structure.

  The attention mechanism enhances the model's focus on transaction patterns, enabling it to prioritize relationships that are more likely to indicate fraud, such as suspicious sending or receiving behaviors.

3.5.2.4 Inter-Layer Fusion and Transformer

After processing the graph through multiple GNN layers, the model aggregates the transaction node embeddings across layers to capture multi-scale relational information.

Inter-Layer Fusion The transaction node embeddings from each GNN layer are concatenated along the feature dimension, resulting in a high-dimensional representation (e.g., 128 dimensions for 2 layers with 64-dimensional embeddings). This fusion captures both local and higher-order patterns, as early layers focus on immediate neighbors, while deeper layers incorporate broader network contexts.

Transformer Encoder The concatenated embeddings are processed by a transformer encoder to model sequential or contextual dependencies among transaction nodes. The transformer consists of a single encoder layer with multiple attention heads (e.g., 4 heads) and a feed-forward network. The input embeddings are treated as a sequence (with a singleton batch dimension), and the transformer applies self-attention to refine the representations:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V,$$

where Q, K, and V are query, key, and value matrices derived from the input embeddings, and d_k is the dimension of each attention head. The transformer output is squeezed to remove the batch dimension, producing refined transaction embeddings that capture both graph-based and sequential patterns.

3.5.2.5 Final Classification

The refined transaction embeddings are passed through a multi-layer perceptron (MLP) for final classification. The MLP consists of two linear layers with a ReLU activation in between:

$$h_{\text{out}} = \text{ReLU}(W_1 h + b_1), \quad \text{logits} = W_2 h_{\text{out}} + b_2,$$

where W_1 and W_2 are weight matrices, and b_1 and b_2 are biases. The output logits have a dimension of 2, corresponding to the binary classification task (fraudulent or non-fraudulent). The logits can be passed through a softmax function during inference to obtain probabilities.

3.6 Model Architecture

The first model leverages the Graph Attention Network (GAT), which incorporates attention mechanisms to weigh node neighborhood contributions. The second model, a Spatial-Temporal Aware Graph Transformer (STGAT), is tailored for heterogeneous graphs and integrates temporal dynamics along with structural information. Below, we describe the architecture of each model in detail.

3.6.1 Graph Attention Network Model

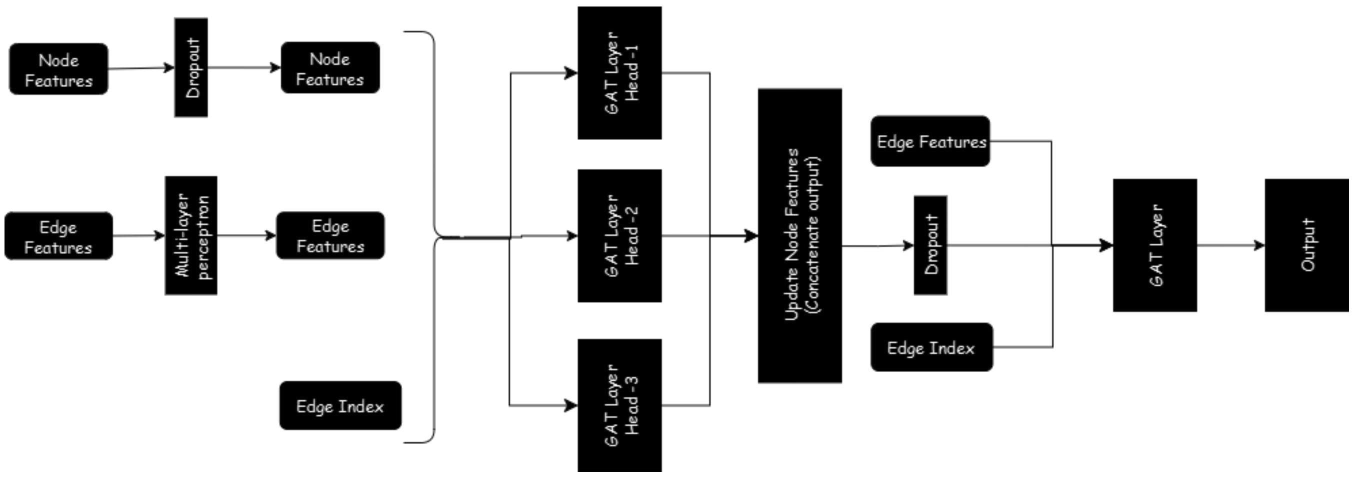


Figure 3.6: Architecture of the Graph Attention Model, integrating node and edge features through two GATConv layers with an edge feature MLP for graph-based learning.

3.6.2 Spatial Temporal Aware Graph Transformer

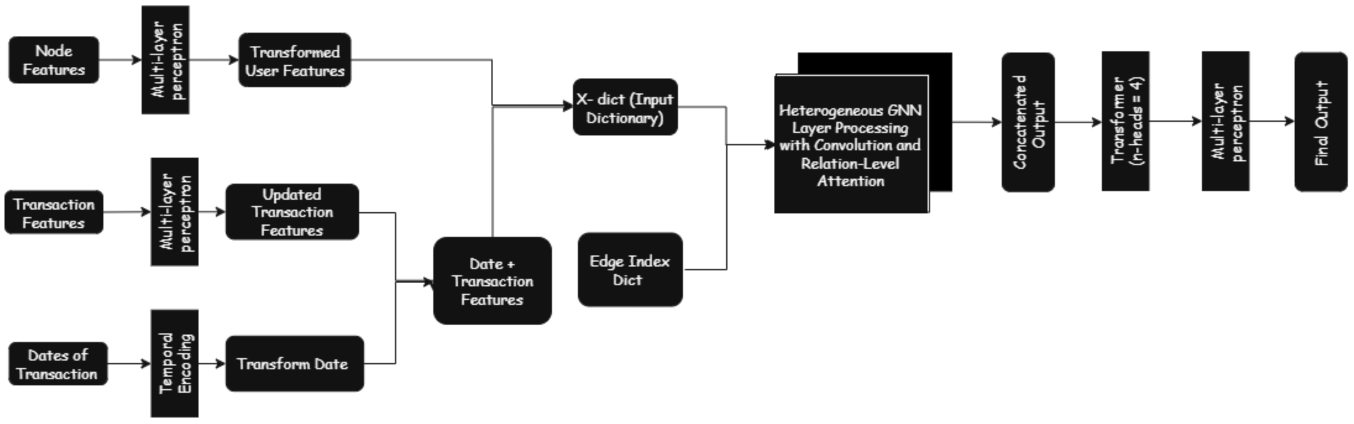

Node Features
Multi-layer perceptron
Transformed User Features
Transaction Features
Multi-layer perceptron
Updated Transaction Features
Dates of Transaction
Temporal Encoding
Transform Date
Date + Transaction Features
X- dict (Input Dictionary)
Edge Index Dict
Heterogeneous GNN Layer Processing with Convolution and Relation-Level Attention
Concatenated Output
Transformer (n-heads = 4)
Multi-layer perceptron
Final Output

Figure 3.7: Flowchart of the Heterogeneous Graph Neural Network Model for Fraud Detection. The model processes a graph of user and transaction nodes, applying initial feature embedding with temporal encoding, iterative heterogeneous GNN layers with relation-level attention, inter-layer fusion, transformer encoding, and a final MLP for binary fraud classification.

3.7 Training

To effectively detect fraudulent activities within financial transaction networks, we trained two different graph-based models: Graph Attention Network (GAT) and Spatial-Temporal Graph Attention Network (STGAT). Each model was trained on a distinct type of graph dataset, tailored to exploit the structural and temporal patterns inherent in financial data. The training methodologies were adapted to account for challenges such as class imbalance, temporal ordering of transactions, and scalability to large graphs. Below, we describe the training procedures and configurations used for each model in detail.

3.7.1 GAT

The model was trained on a graph dataset designed for bank fraud detection, comprising node features, edge indices, and edge attributes, loaded from a preprocessed file. The dataset was split into 70% training and 30% testing nodes using random masking to ensure robust evaluation. The model, configured with 6 input channels, 16 hidden channels, 2 output channels for binary classification, and 3 attention heads, was initialized.
Training was conducted over 100 epochs using the Adam optimizer with a learning rate of 0.005 and a weight decay of 5e-4 to prevent overfitting. A class-weighted cross-entropy loss was employed to address class imbalance, with weights set to 1.0649 and 16.4036 for the respective classes, reflecting the skewed distribution of fraud and non-fraud cases. During each epoch, the model processed the entire graph, applying the loss computation solely to the training nodes. Model performance was evaluated after each epoch on the test set, calculating accuracy based on predictions for test nodes. The training loop tracked the loss and test accuracy, printing these metrics for each epoch to monitor progress. The best test accuracy was recorded, and final metrics, including recall and a confusion matrix, were computed post-training.

3.7.2 STGAT

The training process is designed to optimize the model for detecting fraudulent transactions within a heterogeneous graph dataset. The dataset, comprising transaction and user nodes with associated features, edges, and temporal information, is preprocessed to ensure compatibility with the model. The transaction features, initially with 8 dimensions, and user features, with 31 dimensions. Transaction dates are transformed into timestamps, enabling temporal encoding, and labels indicating whether transactions are fraudulent (1) or legitimate (0).
A time-based split divides the dataset into 70% training and 30% testing sets, sorted by transaction timestamps to maintain temporal consistency. The training set contains a significant class imbalance, with legitimate transactions vastly outnumbering fraud. To address this, class weights are calculated based on the inverse frequency of each class in the training set, ensuring the model prioritizes learning from the minority class (fraudulent transactions). These weights are incorporated into the loss function to balance the training process.
The model is trained using a specialized data loader, named HGTLoader, that samples subgraphs centered on transaction nodes, facilitating efficient batch processing. Each batch

includes up to 4096 transaction nodes, with two layers of neighbor sampling (2048 neighbors per layer), ensuring the model captures relevant graph structure. The training loop iterates over these batches for a single epoch, with the model set to training mode. The optimizer's gradients are reset for each batch, and the model performs a forward pass to generate predictions. The weighted cross-entropy loss is computed between the predictions and true labels, backpropagated, and the optimizer updates the model parameters using a learning rate of 0.01.

Chapter 4

Evaluation and Result Analysis

4.1 Evaluation Metrics

The evaluation methodology and result analysis for two Graph Neural Network (GNN) models—Edge-Enhanced Graph Attention Network (GAT) and Spatio-Temporal Graph Attention Network (STGAT)—developed for fraud detection in financial transaction networks. The models were trained and evaluated on a graph dataset to classify transactions as legitimate or fraudulent, addressing the challenge of imbalanced classes in financial fraud detection.

4.1.1 Evaluation Methodology

The models were evaluated on separate test sets to assess their performance in detecting fraudulent transactions. The evaluation metrics used are:

1. **Confusion Matrix**: A matrix showing true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) to provide a detailed view of classification performance.
2. **Recall**: The ratio of correctly identified fraudulent transactions to all actual fraudulent transactions, critical for minimizing missed fraud cases.
3. **Accuracy**: The proportion of correctly classified transactions (legitimate and fraudulent), indicating overall performance.
4. **Area Under the ROC Curve (AUC)**: The AUC measures the model's ability to distinguish between legitimate and fraudulent transactions across various classification thresholds, with higher values indicating better discriminative performance.

4.1.2 GAT Evaluation Methodology

The GAT model was evaluated on a homogeneous graph where nodes represent transactions and edges denote relationships. The dataset was randomly split into 70% training and 30% testing nodes using a boolean mask. The model was trained for 100 epochs with class weights ([1.0649, 16.4036]) to address class imbalance, using the Adam optimizer (learning rate: 0.005, weight decay: 5×10^{-4}) and cross-entropy loss. For evaluation, the model performed a forward pass on the entire graph, and predictions were extracted for test nodes only. The confusion matrix, recall, and accuracy were computed by comparing predicted labels to true labels on the test set. Additionally, the ROC curve was generated using the predicted probabilities for the positive class, and the AUC was calculated to assess the model's discriminative ability. This approach ensures that the model's performance is assessed on unseen nodes while leveraging the full graph structure during inference.

4.1.3 STGAT Evaluation Methodology

The STGAT model was evaluated on a heterogeneous graph comprising transaction and user nodes, with temporal information incorporated via transaction timestamps. The dataset was split chronologically, with the earliest 70% of transactions used for training and the latest 30% for testing, mimicking real-world fraud detection where future transactions are predicted. The model was trained for one epoch using the HGTLoader for batch processing (batch size: 4096), with dynamically calculated class weights based on the training set's label distribution. The Adam optimizer (learning rate: 0.01) and weighted cross-entropy loss were employed. During evaluation, the test set was processed in batches (batch size: 2048) to handle large-scale graphs. For each batch, a subgraph was created including test transaction nodes and their connected user nodes (1-hop neighbors). Predictions were made on these subgraphs, and the confusion matrix, recall, and accuracy were calculated by aggregating predictions and true labels across all batches. The ROC curve was constructed using the predicted probabilities for the positive class from all batches, and the AUC was computed to evaluate the model's ability to distinguish between classes. Additionally, a harmonic mean (HM) score of recall and accuracy, defined as

$$\text{Score} = \frac{2 \cdot \text{recall} \cdot \text{accuracy}}{\text{recall} + \text{accuracy} + 10^{-8}} \tag{4.1}$$

was used to select the best model checkpoint, balancing fraud detection and overall performance. This batched evaluation ensures scalability while preserving the heterogeneous and temporal context.

4.2 Result Analysis

The results for GAT and STGAT are presented below, and these results are on the subset dataset, it contain total of 157968 transaction samples. The analysis focuses on the confusion matrix, recall, accuracy, and AUC, with the harmonic mean score considered for STGAT.

4.2.1 GAT Results

The GAT model achieved the following performance on the test set:

1. **Confusion Matrix**: The confusion matrix is shown in Table 4.1, indicating the distribution of classifications. The matrix reveals a high number of true negatives (39,037)

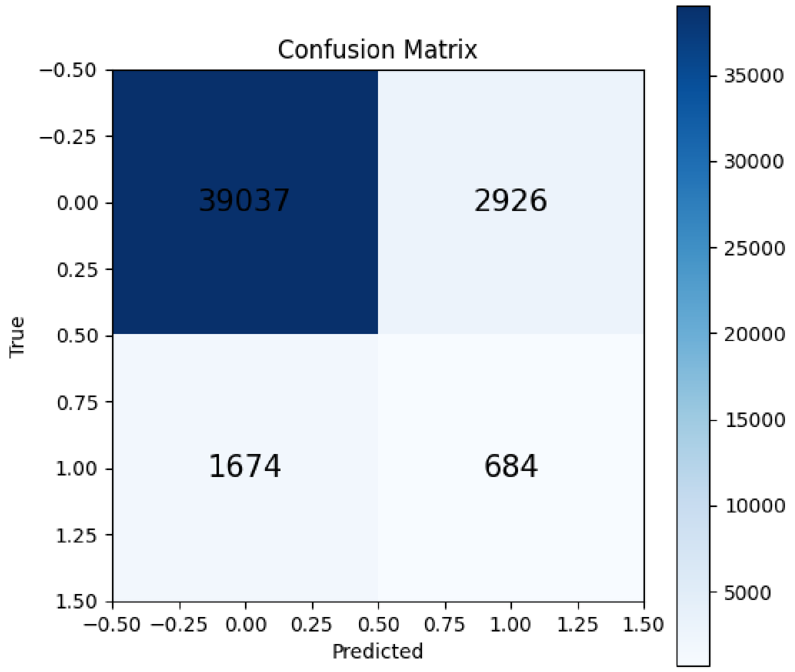


Figure 4.1: Confusion Matrix for GAT

   but a significant number of false negatives (1,674), suggesting challenges in detecting all fraudulent transactions. The false positives (2,926) indicate some legitimate transactions were misclassified as fraudulent.

2. **Recall**: The recall is 0.2901, meaning 29.01% of fraudulent transactions were correctly identified. This moderate recall highlights the model's difficulty in capturing the majority of fraud cases, likely due to class imbalance.

3. **Accuracy**: The accuracy is 0.8962 (89.62%), reflecting strong overall performance but potentially skewed by the dominance of legitimate transactions (true negatives).

4. **AUC**: The ROC curve, shown in Figure 4.2, illustrates the model's discriminative ability, with an AUC of 0.61. A higher AUC would indicate better separation between classes, but the moderate recall suggests limitations in distinguishing fraudulent transactions.

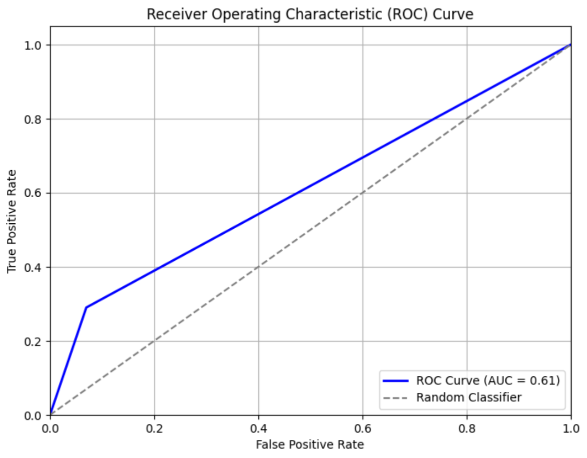


Figure 4.2: ROC Curve for GAT (AUC = 0.61).

4.2.2 STGAT Results

The STGAT model achieved the following performance on the test set:

1. **Confusion Matrix**: The confusion matrix is shown in Table 4.3, indicating the distribution of classifications.

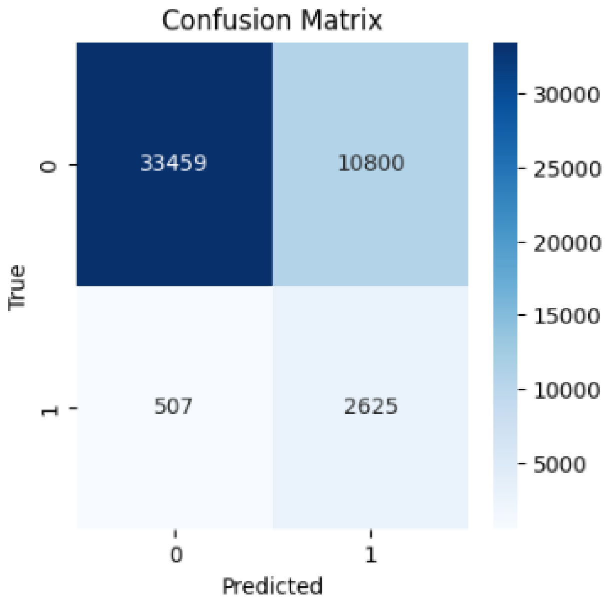


Figure 4.3: Confusion Matrix for GAT

The matrix shows a high number of true negatives (33,459) and true positives (2,625), with relatively few false negatives (507), indicating strong fraud detection. However, the high false positives (10,800) suggest many legitimate transactions were misclassified as fraudulent.

2. **Recall**: The recall is 0.8381, meaning 83.81% of fraudulent transactions were correctly identified. This high recall demonstrates STGAT's effectiveness in capturing most fraud cases, likely benefiting from its heterogeneous and temporal modeling.
3. **Accuracy**: The accuracy is 0.7614 (76.14%), indicating solid overall performance.
4. **AUC**: The ROC curve, shown in Figure 4.4, illustrates the model's discriminative ability, with an AUC of 0.8865. This matches GAT's AUC, suggesting comparable separation between classes.

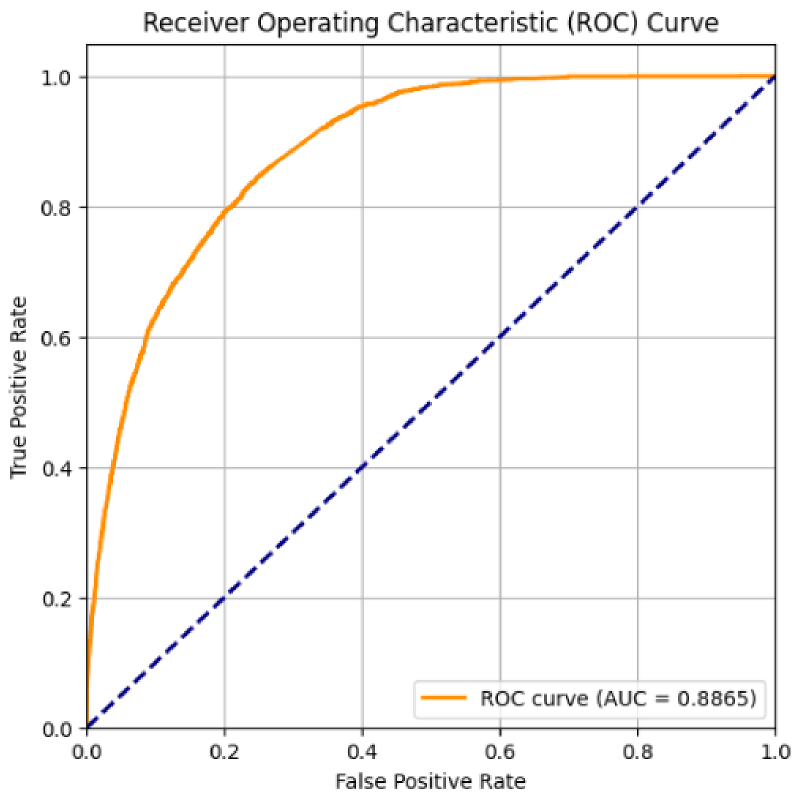


Figure 4.4: ROC Curve for STGAT (AUC = 0.8865).

5. **Harmonic Mean (HM) Score**: The HM score of recall and accuracy is 0.7979, reflecting a strong balance between fraud detection (recall) and overall performance (accuracy). This score was used to select the best model checkpoint, indicating a model optimized for practical fraud detection.
6. **Loss Plot (STGAT)**: The training loss for STGAT is shown in Figure 4.5, illustrating the model's convergence behavior.

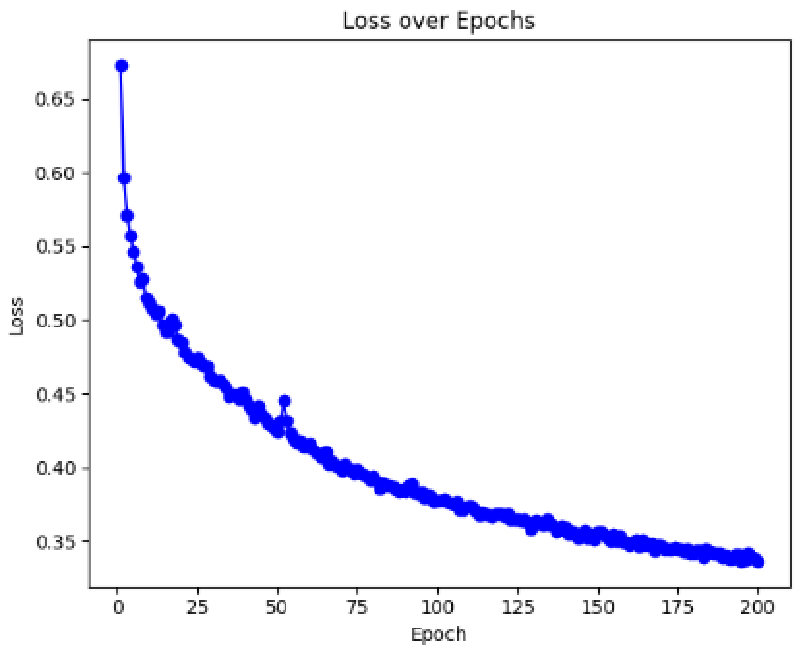


Figure 4.5: Training Loss Curve for STGAT

4.3 Comparative Insights

The comparison of GAT and STGAT based on the specified metrics reveals the following insights:

1. **Fraud Detection Effectiveness (Recall)**: STGAT significantly outperforms GAT in recall (0.8381 vs. 0.2901), correctly identifying 83.81% of fraudulent transactions compared to only 29.01%. The confusion matrices highlight this disparity: STGAT has far fewer false negatives (507 vs. 1,674), indicating superior fraud detection. However, STGAT's higher false positives (10,800 vs. 2,926) suggest it flags more legitimate transactions as fraudulent, which may increase investigation costs.
2. **Overall Performance (Accuracy)**: GAT achieves higher accuracy (0.8962 vs. 0.7614), correctly classifying 89.62% of transactions compared to STGAT's 76.14%. This is largely due to GAT's high true negatives (39,037 vs. 33,459), reflecting its effectiveness on the majority class (legitimate transactions). However, this high accuracy is skewed by class imbalance and less relevant for fraud detection, where recall is critical.
3. **Discriminative Ability (AUC)**: STGAT demonstrates superior discriminative ability with an AUC of 0.8851 compared to GAT's 0.61, as shown in Figures 4.2 and 4.4. STGAT's higher AUC indicates better separation between legitimate and fraudulent transactions across various thresholds, aligning with its higher recall and effectiveness in fraud detection.

4. **Confusion Matrix Analysis**: GAT's confusion matrix (Table 4.1) shows a high number of true negatives but significant false negatives (1,674), limiting its fraud detection capability. STGAT's matrix (Table 4.3) demonstrates strong fraud detection with 2,625 true positives and only 507 false negatives, but its 10,800 false positives indicate a trade-off in misclassifying legitimate transactions. Heatmaps of these matrices (Figures 4.1 and 4.3) visually confirm these patterns.

4.4 Results on Complete Dataset

This section presents the performance of a model trained for fraud detection on a Complete Dataset of financial transactions. The dataset comprises 9,504,852 transactions, with 9,494,979 non-fraudulent (99.9%) and 9,873 fraudulent (0.1%) transactions, reflecting severe class imbalance. The model's performance is evaluated using four key metrics: confusion matrix, recall, accuracy, and Area Under the Receiver Operating Characteristic Curve (AUC), with the harmonic mean (HM) score of recall and accuracy included as an additional metric.

4.4.1 Performance Metrics

The performance of model on the test set is summarized below, with the confusion matrix presented in Table **??**, the ROC curve in Figure **??**, and the training loss curve in Figure **??**.

1. **Confusion Matrix**: The confusion matrix (Table 4.6) shows the distribution of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), pending specific values.

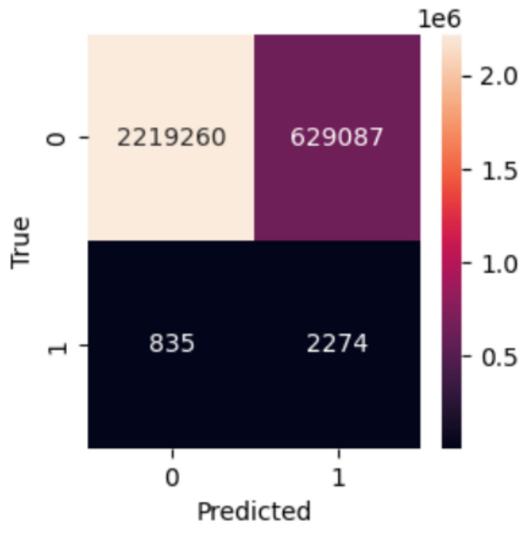


Figure 4.6: Confusion Matrix for STGAT

The matrix indicates the model's ability to correctly classify fraudulent and non-fraudulent transactions. A low number of false negatives (FN) would suggest effective fraud detection, while false positives (FP) reflect misclassified legitimate transactions, impacting operational efficiency.

2. **Loss Plot (STGAT)**: The training loss for STGAT in complete dataset is shown in Figure 4.5, illustrating the model's convergence behavior.

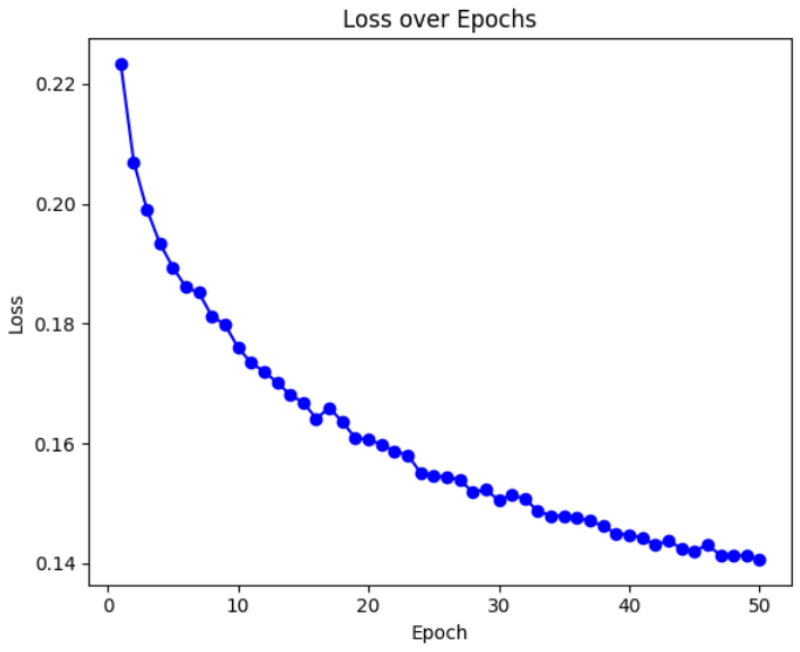


Figure 4.7: Training Loss Curve for STGAT on Complete Dataset

3. **Recall**: 0.7314, indicating that 73.14% of fraudulent transactions were correctly identified. This strong recall demonstrates the model's effectiveness in capturing most fraud cases, critical for minimizing missed fraudulent transactions.

4. **Accuracy**: 0.7791, indicating that 77.91% of all transactions were correctly classified. This accuracy reflects solid overall performance, though it may be influenced by the dataset's class imbalance.

5. **Harmonic Mean (HM) Score**: 0.7545, reflecting a balanced performance between fraud detection (recall) and overall classification (accuracy).

6. **AUC**: 0.8400, indicating the model's discriminative ability to distinguish between fraudulent and non-fraudulent transactions, as shown in the ROC curve (Figure 4.8). A higher AUC would reflect better separation across classification thresholds.

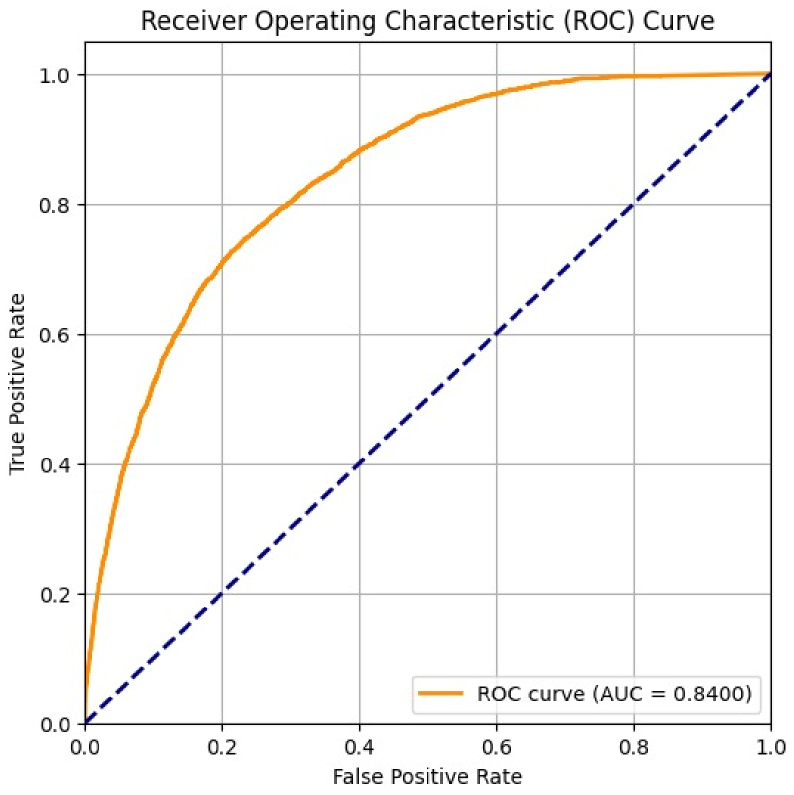


Figure 4.8: ROC Curve for STGAT on Complete Dataset(AUC = 0.8400)

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The financial sector faces increasingly sophisticated fraud schemes that involve complex transactional networks and patterns difficult to detect with traditional methods. This research successfully addresses this challenge by leveraging Graph Neural Networks, particularly Graph Attention Networks (GATs) and Spatial-Temporal Graph Attention Networks (STGATs), to model and analyze financial transaction networks where accounts represent nodes and transactions form edges.

The research journey involved several crucial stages that contributed to its effectiveness. First, extensive data preprocessing transformed raw transaction data into appropriate graph structures, addressing challenges such as class imbalance (only 0.1% of transactions being fraudulent) and feature heterogeneity. Two distinct graph construction approaches were implemented: a homogeneous graph modeling transactions between accounts, and a heterogeneous graph incorporating temporal dynamics by distinguishing between user and transaction nodes.

The comparative evaluation of GAT and STGAT models revealed significant insights into their fraud detection capabilities. The STGAT model substantially outperformed the GAT model in recall (83.81% vs. 29.01%) and discriminative ability (AUC of 0.8865 vs. 0.61), demonstrating its superior effectiveness at identifying fraudulent transactions. This performance difference highlights the value of incorporating temporal information and heterogeneous graph structures when modeling financial transactions. Although GAT achieved higher overall accuracy (89.62% vs. 76.14%), this metric is less relevant in the context of fraud detection where identifying the minority fraudulent class is paramount.

When applied to the complete dataset of over 9.5 million transactions, the STGAT model maintained robust performance with a recall of 73.14%, accuracy of 77.91%, and an AUC of 0.84. This demonstrates the model's scalability to real-world transaction volumes while maintaining effective fraud detection capabilities. The model's harmonic mean score of 0.7545 reflects a balanced performance between fraud detection sensitivity and overall classification accuracy.

One particularly valuable aspect of the attention mechanism in both models is its ability to assign varying importance to different neighboring nodes, allowing the models to focus on the most suspicious transaction patterns. This dynamic weighting proves especially valuable in financial fraud detection, where certain transaction relationships are more indicative of fraudulent behavior than others. The edge-featured enhancement of the GAT model further improved its performance by incorporating transaction characteristics into the attention mechanism, boosting the True Positive Rate to 69% while maintaining high precision of 93%.

Despite these promising results, challenges remain. The STGAT model's increased false positive rate (10,800 false positives compared to GAT's 2,926) indicates potential operational challenges in real-world deployment, where investigating large numbers of legitimate transactions flagged as fraudulent could strain resources. Additionally, the current approach focuses on node classification rather than subgraph analysis, potentially missing complex fraud schemes spanning multiple accounts and transactions.

5.2 Future Work

Building upon the current research findings, several promising directions for future work emerge that could enhance banking fraud detection capabilities using graph-based approaches.

5.2.1 Advanced Model Architectures and Ensemble Methods

To improve the True Positive Rate beyond the current performance level of 69%, future research should explore more sophisticated GNN architectures. Specifically, integrating Graph Convolutional Networks (GCNs) with Graph Isomorphism Networks (GINs) could enhance the model's ability to capture both local and global structural patterns in transaction networks. Additionally, implementing hierarchical attention mechanisms would allow the model to focus on multiple levels of the graph simultaneously, from individual nodes to transaction neighborhoods and broader community structures.

Ensemble methods combining multiple graph-based models could further improve performance. A potential approach involves implementing a stacking ensemble that leverages the strengths of different GNN architectures – using GAT for identifying suspicious individual accounts, GraphSAGE for scalable neighborhood sampling in large transaction networks, and STGAT for capturing temporal patterns. This multi-model approach could be particularly effective in addressing the diverse fraud patterns observed in financial networks.

5.2.2 Subgraph-Level Detection and Classification

Current node-level classification approaches miss critical insights available in transaction chains and communities. Future work should develop methods for classifying entire transaction trails (subgraphs) as fraudulent or legitimate, which would better capture complex money laundering schemes that distribute suspicious activities across multiple accounts to avoid detection.

Implementing algorithms for community detection within transaction networks could identify clusters of accounts working together in coordinated fraud schemes. Graph pooling techniques, similar to those used in image recognition, could be adapted to aggregate node-level features into meaningful subgraph representations that preserve structural information while enabling classification at the subgraph level. This would allow the detection of fraud patterns that emerge only when considering groups of related transactions rather than individual ones.

5.2.3 Generative Models for Transaction Trail Reconstruction

An innovative direction for future research involves developing generative models capable of reconstructing incomplete transaction trails. In real-world scenarios, financial transaction data often contains gaps due to missing records or limitations in data collection processes. These gaps can obscure critical patterns in complex fraud schemes spanning multiple accounts.

Graph generative models, such as Graph Variational Autoencoders (GVAEs) or Graph Generative Adversarial Networks (GraphGANs), could be trained to predict missing links in transaction networks based on existing structural patterns and account behaviors. The model could learn to infer plausible paths between disconnected components in the transaction graph, revealing potential fraud schemes that might otherwise remain hidden due to data incompleteness.

Additionally, these generative approaches could be extended to simulate potential future transaction patterns, enabling proactive fraud detection by identifying account behavior that is likely to lead to fraudulent activities before they occur. This predictive capability would represent a significant advancement over current reactive detection methods.

5.2.4 Real-Time Detection and Adaptive Learning

To address the evolving nature of financial fraud, future research should explore online learning techniques that enable models to continuously update based on new transaction data and confirmed fraud cases. This would allow the system to adapt to emerging fraud patterns without requiring complete retraining.

Implementing a feedback loop where fraud investigators' confirmations or rejections of system alerts are used to refine the model would create a continuously improving detection system. Combining this with reinforcement learning approaches could optimize the trade-off between recall and false positive rates, dynamically adjusting the model's sensitivity based on operational constraints and the evolving fraud landscape.

The integration of these advanced approaches would significantly enhance the effectiveness of graph-based fraud detection systems, moving beyond individual transaction analysis to comprehensive network-level understanding of fraudulent activities in financial systems.

Bibliography

[1] Abdul Joseph Fofanah, David Chen, L. W. S. Z. [2023], 'Addressing imbalance in graph datasets: : Introducing gate-gnn with graph ensemble weight attention and transfer learning for enhanced node classification', **255**.

[2] Cheng, D., Wang, X., Zhang, Y. and Zhang, L. [2022], 'Graph neural network for fraud detection via spatial-temporal attention', *IEEE Transactions on Knowledge and Data Engineering* **34**(8), 3800–3813.

[3] Hamilton, W. L., Ying, R. and Leskovec, J. [2018], 'Inductive representation learning on large graphs'.
URL: *https://arxiv.org/abs/1706.02216*

[4] Jie Zhou, Ganqu Cui, S. H. Z. Z. C. Y. Z. L. L. W. C. L. M. S. [2019], 'Graph neural networks: A review of methods and applications', *AI open* **5**.

[5] Jun Chen, H. C. [2021], 'Edge-featured graph attention network', *cs.LG* **1**.

[6] Khemani, B., Patil, S., Kotecha, K. and Tanwar, S. [2024], 'A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions', *Journal of Big Data* **11**.

[7] Liyu Gong, Q. C. [2019], 'Exploiting edge features in graph neural networks : This explores edge feature enhancements in gnns, including applications in attention mechanisms', *cs.LG* **2**.

[8] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. and Monfardini, G. [2009], 'The graph neural network model', *IEEE Transactions on Neural Networks* **20**(1), 61–80.

[9] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I. [2023], 'Attention is all you need'.
URL: *https://arxiv.org/abs/1706.03762*

[10] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P. and Bengio, Y. [2018], 'Graph attention networks', *ICLR* **1**.

[11] Zhang, C., Song, D., Huang, C., Swami, A. and Chawla, N. V. [2019], 'Heterogeneous graph neural network'.
URL: *https://doi.org/10.1145/3292500.3330961*

[12] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C. and Sun, M. [2020], 'Graph neural networks: A review of methods and applications', *AI Open* **1**, 57–81.

[13] Ziniu Hu, Yuxiao Dong, K. W. Y. S. [2020], 'Heterogeneous graph transformer : This paper introduces extensions to gat for heterogeneous graphs, including edge-specific enhancements.', *cs.LG* **1**.

20EC39024_Report.pdf

ORIGINALITY REPORT

16%	8%	11%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Indian Institute of Technology, Kharagpure Student Paper	1%
2	arxiv.org Internet Source	1%
3	Submitted to Liverpool John Moores University Student Paper	<1%
4	dokumen.pub Internet Source	<1%
5	Submitted to UNESCO-IHE Institute for Water Education Student Paper	<1%
6	Soroor Motie, Bijan Raahemi. "Financial fraud detection using graph neural networks: A systematic review", Expert Systems with Applications, 2024 Publication	<1%
7	Submitted to University of Hong Kong Student Paper	<1%
8	Submitted to Heriot-Watt University Student Paper	<1%
9	www.mdpi.com Internet Source	<1%
10	Submitted to University of Bristol Student Paper	<1%
11	Schwab, Isaac. "Solving Large Job Shop Scheduling Problems: Using Graph Classification via Graph Neural Networks to	<1%

	Pre-Seed a Genetic Algorithm for Machine Dispatching Rule Optimization", The University of Wisconsin - Milwaukee, 2024 Publication	
12	Samia Saidane, Francesco Telch, Kussai Shahin, Fabrizio Granelli. "Deep GraphSAGE enhancements for intrusion detection: Analyzing attention mechanisms and GCN integration", Journal of Information Security and Applications, 2025 Publication	<1%
13	Aristidis G. Vrahatis, Konstantinos Lazaros, Sotiris Kotsiantis. "Graph Attention Networks: A Comprehensive Review of Methods and Applications", Future Internet, 2024 Publication	<1%
14	Abdul Joseph Fofanah, David Chen, Lian Wen, Shaoyang Zhang. "CHAMFormer: Dual heterogeneous three-stages coupling and multivariate feature-aware learning network for traffic flow forecasting", Expert Systems with Applications, 2024 Publication	<1%
15	Chun-Hu Pan, Yi Qu, Yao Yao, Mu-Jiang-Shan Wang. "HybridGNN: A Self-Supervised Graph Neural Network for Efficient Maximum Matching in Bipartite Graphs", Symmetry, 2024 Publication	<1%
16	propulsiontechjournal.com Internet Source	<1%
17	boa.unimib.it Internet Source	<1%
18	core.ac.uk Internet Source	<1%
19	ceur-ws.org Internet Source	<1%