Vrije Universiteit Amsterdam

Avanade

Master Thesis

What If we Cannot See the Full Picture?

Anti-Money Laundering in Transaction Monitoring

Author: Sara Lute (2785832)

1st supervisor:	Professor A.E Eiben	
daily supervisor:	Paulien Leeuwenburgh	(Avanade)
2nd reader:	Doctor K.S. Luck	

A thesis submitted in fulfillment of the requirements for the VU Master of Science degree in Business Analytics

July 19, 2024

Abstract

In recent years, Dutch banks have faced significant scrutiny and substantial fines from De Nederlandsche Bank (DNB) for failing to comply with the Dutch Money Laundering and Terrorist Financing Prevention Act (Wwft). Wwft requires financial institutions to conduct customer research and report unusual transactions to the Financial Intelligence Unit (FIU) of the Netherlands. However, current anti-money laundering (AML) systems, which rely on outdated rule-based models, struggle to keep pace with evolving money laundering tactics. These systems generate numerous false positives, leading to costly and time-consuming manual investigations. Although machine learning models show promise for improved transaction monitoring, their implementation is hindered by large fines and limited data access due to privacy regulations.

This paper presents a comparative research on anti-money laundering (AML) models in transaction monitoring with incomplete graph networks. The goal of this research is to answer the question "What can we do when we cannot see the full picture?" and to develop an effective approach for handling missing nodes in partial networks for AML transaction monitoring.

Various Graph Neural Networks are trained with different adaptions on both the full dataset with complete money laundering networks and subsets of the dataset with only the transactions accesible to a single bank. Experimental results demonstrate that the proposed model improves the performance on AML tasks in the case of incomplete data. The findings emphasise that sophisticated machine learning algorithms, while promising, require comprehensive transaction data to function optimally. The research suggests the exploration of anonymised data-sharing methods and advanced temporal modelling techniques to improve the robustness and effectiveness of AML transaction monitoring systems in the real world.

Keywords: anti-money laundering (AML) & transaction monitoring & graph convolutional networks (GCNs) & temporal graph networks & incomplete graph networks & privacy regulations & false positives & machine learning models & laundering patterns & data sharing

Table of Contents

List of Figures

List of Tables

1 Introduction

In recent years, Dutch banks have been under investigation by De Nederlandsche Bank (DNB) and have received heavy fines due to not living up to the Dutch Money Laundering and Terrorist Financing Prevention Act (Wet ter voorkoming van witwassen en financieren van terrorisme (Wwft[50])) (e.g., ING[7], ABN[3], RaboBank[5], Volksbank[6]).

According to Wwft, institutions and professional groups involved in money flows or in the purchase and sale of goods[30] must perform customer research and report unusual transactions to the Financial Intelligence Unit (FIU) of the Netherlands[36]. Customer research entails identifying the customer, checking whether the customer represents, and if so, is allowed to represent someone, identifying the customer's ultimate beneficial owner, identifying the customer's purpose, and checking whether the customer is making unusual transactions (transaction monitoring).

This last task is a growing bottleneck in this industry due to the lack of a proper anti-money laundering (AML) model for transaction monitoring. They point out that the current systems at the large Dutch banks are 30-years-old rule-based models which have become outdated, since until recently there was no control/consequences on the money laundering screening.

The outdated transaction monitoring models currently flag fraudulent behaviour based on rules defined by a team of domain specialists. Changing an existing rule or creating a new one takes months; however, money launderers change tactics daily. Since De Nederlandse Bank (DNB) started monitoring money laundering screening more closely, many large fines have been issued. Current models lead to too many incorrect labels, especially many false positives, due to the need to cover all true positives. This is very costly since the banks get reprimanded for incorrect labelling, but besides that each positive, false or not, is manually investigated in a time-consuming process. Machine learning models show promise for faster adaption to new money laundering tactics and offer a potential reduction of false positives. However, due to the large fines, banks have become hesitant to implement any big changes, such as the relatively unexplored implementation of machine learning models.

In addition to this, the limited access to the complete set of transaction data is a problem. The quality of a machine learning model relies heavily on the input data, and with the current privacy regulations in the Netherlands, it is not possible for one entity, such as Transaction Monitoring Nederland (TMNL), to access all transactions. TMNL is a collaboration between five major Dutch banks to fight financial crime by monitoring bank transactions of banks together for money laundering. For this initiative to succeed, updates to Dutch legislation are necessary. Under the current legal framework, TMNL limits its use of data and uses pseudonymization to protect privacy, in accordance with GDPR guidelines, focussing exclusively on monitoring business transactions. For multiple reasons, such as the fall of the Dutch cabinets and the opposition of the Authoriteit Persoonsgegevens (AP) and Brussels[4], it will not be feasible to wait for a party (like TMNL) to have access to the full picture of the data in which money launderers operate[1].

Money laundering is an elusive activity. A money launderer will adapt and innovate their tactics. As explained by Schneider and Windischbauer[40], there are different phases of money laundering, namely, the placement, layering, and integration phases. During placement, illicit funds are deposited into the financial system, often in small increments to avoid detection. In the layering phase, these funds are moved through various transactions, creating a web of financial activity that masks the original source. Finally, the integration phase sees the laundered money re-enter the economy as legitimate through investments in businesses or properties, completing the money laundering cycle. Money launderers specifically choose to use this layering phase due to the lack of / easily bypassed know your customer (KYC[35]) and customer due diligence procedures, and because transfers between jurisdictions are not reportable[38].

[1] Towards the end of this research (July 1st, 2024) new EU regulations went into action. These regulations completely limit the attempts of TMNL [46], forcing them to redesign their business plan.

The problem is thus rooted in two circumstances:

1. We can only consider transaction data gathered within one bank. (Simplified reason: banks do not have access to transactions of other banks, and no third party has all data).
2. A money launderer does not do its entire business within one bank. For example, fraudulent transactions will be funnelled through accounts at different banks.

Contribution. The overall goal of this research is to answer the question "What if we cannot see the full picture?". The technical goal is to explore approaches to handle missing information in graph networks for AML transaction monitoring. In this research, adaptions to existing Graph Neural Network (GNN) approaches are proposed, explicitly modelling the temporal and incomplete information components in the networks.

Organization. Section 1 introduces the topic and describes the key objectives and scope of the research. This is followed by Section 2, which provides a comprehensive review of the existing literature relevant to the study and the previous work at the host company of this research, Avanade. In Section 3, the methods are presented; where Section 3.1 describes the Multi-GNN model from the referenced paper by Egressy et al,[19], which serve as the baseline models, while Section 3.2 details the proposed components addressing the temporal adaptions and Section 3.3 that addresses the adaption for incomplete information. Then, Section 4 dives into the dataset used in this research and how the single-bank view influences the access to the laundering networks in the data. Section 5 outlines the setup and procedure of the experiments. Continuing with Section 6 that describes and evaluates the results. Then, the discussion in Section 7. Finally, the report concludes in Section 8 with the conclusion and proposals for future research.

2 Related work

Money laundering is a criminal process that is used to convert criminal income into assets, disguising the source of illegally obtained money and making it appear legitimate among all other transactions [26]. Money laundering poses a significant threat to the integrity of financial systems because it enables criminals to enjoy the proceeds of their illegal activities while evading detection and prosecution. A paradigm used throughout the literature to explain the money laundering process is the three common phases: placement, layering, and integration [32].

- *Placement*, the first phase, represents the entry of illicit funds into the financial system, for example, with cash deposits, currency exchanges, or high-value asset acquisitions. During this phase, money launderers exploit vulnerabilities in the financial system to disguise or misrepresent the source of the funds.
- *Layering*, the second phase, involves transaction structuring, implementing tactics such as complex transactions across jurisdictions or between accounts, and currency conversions to further obscure the audit trail. The goal of *Layering* is to create a convoluted network of transactions, increasing the difficulty for law enforcement agencies to track illicit funds.
- *Integration*, the third phase, completes the money laundering cycle by reintegrating laundered funds into the economy as legitimate assets. For example, through investments in legitimate businesses or acquisitions of real estate or luxury goods. [37,40]

The problem of money laundering presents a tough challenge that has been addressed by several approaches. The following sections will introduce the key topics for this research. First, previous related literature will be reviewed. Next, relevant work within the host company, Avanade, will be examined. Finally, the research problem will be defined.

2.1 Traditional methods

Anti Money Laundering (AML) methods for transaction monitoring in banks traditionally are rule-based systems that have been extensively researched and implemented. To this day, banks are relying on rule-based systems to locate illicit transactions based on predefined static rules [10]. These predefined rules and thresholds are set by domain experts and regulatory requirements. For example, rules may flag transactions that exceed a certain size, involve high-risk jurisdictions, or exhibit unusual patterns compared to a customer's typical behaviour. However, despite their widespread use, these systems often suffer from significant drawbacks. One major issue is the high rate of false positives generated by rule-based systems, estimated in 2019 to be over 98% [12]. Due to the rigid nature of the rules and thresholds, legitimate transactions can easily be labelled as fraudulent, leading to unnecessary and costly manual investigations. For example, a large one-time transaction, such as a bonus, may be incorrectly flagged as suspicious. Moreover, rule-based systems struggle to adapt to the ever-evolving tactics used by money launderers. Since criminals continually come up with new methods to launder illicit funds, static rules can quickly become outdated and ineffective in detecting money laundering. This lack of adaptability makes it difficult for financial institutions to stay ahead of sophisticated financial crimes. Amplifying these challenges is the volume of transactions processed by financial institutions, which can overwhelm manual review processes of (false) positives and makes it easier to hide genuine suspicious activities. The need to balance the need to cover all potential fraud cases and minimise false positives presents a significant challenge [26]. In general, while traditional rule-based systems have been a cornerstone of AML efforts in banks, their limitations in terms of high false positives and lack of adaptability highlight the need for more advanced and dynamic transaction monitoring approaches.

2.2 Supervised learning algorithms

To address the limitations of traditional rule-based systems in transaction monitoring for AML, researchers have increasingly turned to machine learning and data mining techniques. Supervised learning algorithms can be powerful tools for identifying anomalies based on labelled historical data. These models are adaptable, as they can continuously learn from new data to improve their performance over time. By analysing historical transaction data labelled as fraudulent or legitimate, these algorithms can identify patterns and features indicative of suspicious activity [10]. For example, neural networks can capture non-linear relationships within the transaction data, and Support Vector Machines (SVMs) are great at classifying transactions into distinct categories based on their features.

The effectiveness of these models depends heavily on the quality and representativeness of the training data [32]. Biased or incomplete datasets may lead to inaccurate or biased predictions, highlighting the importance of data pre-processing and feature engineering. More importantly, a significant challenge in applying supervised learning methods to transaction monitoring is the scarcity of labelled data [9,14]. Unfortunately, obtaining labelled data, indicating whether a transaction is fraudulent or legitimate, can be challenging due to the relatively low frequency of fraudulent transactions amongst all transactions, let alone those found [24]. Additionally, the existing labelled data quickly becomes outdated as money launderers continuously evolve their techniques to evade detection [52]. As a result, models trained on outdated data may overfit and fail to accurately capture emerging money laundering patterns.

2.2.1 Classification is a supervised learning problem where the response is categorical (”qualitative”). The objective of a classification model is to assign an observation to one of several predefined categories based on the observed features of the data. The model is built by learning from a dataset of examples that have known category labels. Classification models estimate the conditional probability of each category, given the input features, and make predictions by selecting the category with the highest probability for new observations. [21].

2.3 Unsupervised learning algorithms

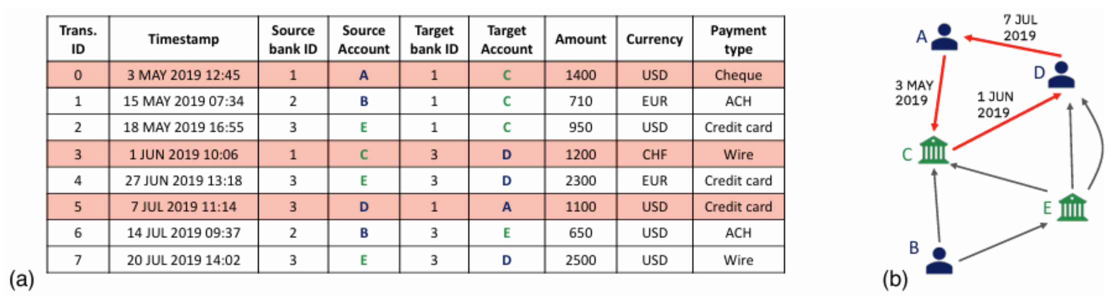
Unsupervised learning methods have emerged as a promising approach to monitoring AML transactions, as an alternative that does not rely on labelled training data. These methods try to identify patterns in the data without information on which data correspond to money laundering and not [24]. Existing research is done, for example, in clustering algorithms and anomaly/outlier detection. Clustering is the most frequently used methodology within unsupervised learning [14]. By grouping transactions based on similarity to uncover clusters of potentially suspicious activity and by analysing transactions within each cluster to identify irregularities or patterns indicative of money laundering. More sophisticated methods do peer-to-peer and peer-to-group comparisons through the clustering process. In this way false positives are reduced when one transaction is unusual given the history of a customer but normal given the common movements of the group with similar characteristics[15]. On the other hand, anomaly detection techniques focus on flagging transactions as suspicious when they do not correlate with most transactional data that are considered normal[32]. These anomalies may signal potential instances of money laundering or other illicit activities. Anomaly detection algorithms are popular in AML; however, not all anomalies are suspicious, which contributes to a high false positive rate. Although struggles with the ability of accurately detecting complex money laundering patterns still depend on the exact model, the problems that arise specifically relevant to AML in transaction monitoring are the reliance on the quality and representativeness of the privacy-regulated transaction data and the explainability of the models.

2.4 Partial networks

As discussed in Section 1, the Dutch data privacy regulations generally limit the sharing and exchange of financial information among institutions. These regulations limit the collaboration between financial institutions, law enforcement agencies, and regulatory bodies in money laundering detection. At the same time, money launderers strategically move their illicit transactions across multiple financial institutions to avoid detection by transaction monitoring systems, as described in the layering phase[40]. By fragmenting their activities, money launderers exploit the oversight gaps created by privacy regulations. The strategic dispersion of illicit transactions in this layering phase amplifies the challenges posed by data privacy regulations, creating an incomplete transaction data network. This increases the risk of missing connections and further complicates efforts to identify suspicious activities. An example of the partial transaction network to which a large Dutch bank would have access is depicted in Figure 3. Existing research into models specifically for AML in transaction monitoring that incorporate components for missing nodes or incomplete networks is limited, where incomplete network information is an assumption but not actively modelled [45]. Models for general learning with partial graph networks offer promising results by either finding the missing nodes first [28,29] or combining the process of finding missing values and forecasting the target value [53,43,27].

2.5 Graphical Transactional Data.

So far all described methods work with tabular data. However, in this research the dataset will be used in graphical form. Graphs offer a natural representation for financial transaction data, revealing the connectivity of underlying data objects and enabling the extraction of complex patterns. In a financial transaction graph, the nodes represent accounts and the directed edges represent financial transactions between these accounts. Multiple transactions between the same two accounts can occur at different times, resulting in directed multigraphs. As depicted in Figure 1, the financial transactions organised in a table format on the left can be reorganised in a graph format, as shown on the right. This graph structure improves the ability to detect suspicious activities by exposing the relationships and patterns within transaction data [1]. Using graphs, it is possible to visualise complex financial networks and identify anomalous behaviour that can indicate fraudulent activity. Furthermore, graph-based representations facilitate the use of advanced machine learning models, such as Graph Neural Networks (GNNs), to analyse and interpret transaction data ([9], [19]).



Trans. ID	Timestamp	Source bank ID	Source Account	Target bank ID	Target Account	Amount	Currency	Payment type
0	3 MAY 2019 12:45	1	A	1	C	1400	USD	Cheque
1	15 MAY 2019 07:34	2	B	1	C	710	EUR	ACH
2	18 MAY 2019 16:55	3	E	1	C	950	USD	Credit card
3	1 JUN 2019 10:06	1	C	3	D	1200	CHF	Wire
4	27 JUN 2019 13:18	3	E	3	D	2300	EUR	Credit card
5	7 JUL 2019 11:14	3	D	1	A	1100	USD	Credit card
6	14 JUL 2019 09:37	2	B	3	E	650	USD	ACH
7	20 JUL 2019 14:02	3	E	3	D	2500	USD	Wire

Fig. 1. Financial transactions in (a) tabular and in (b) graph format. (Altman et al. [1])

2.5.1 Node Embeddings are representations of nodes in a graph in a low-dimensional vector space. The goal of these embeddings is to encode the graph's structural information such that the geometric relationships in the embedding space reflect the graph's topology. This means that nodes that are closer or more similar within the graph structure are also closer in the embedding space. These embeddings facilitate the application of machine learning techniques that require vector inputs by transforming the complex, irregular structures of graphs into a systematic, numerical form that these techniques can process efficiently. [20].

2.5.2 Temporality The temporal aspect of transactions is crucial, especially in tracing laundered money through multiple accounts. Money cannot leave an account before it has arrived, which makes it essential to incorporate temporal dynamics to accurately detect suspicious patterns. An effective method to integrate the temporal variable involves creating subgraphs for each specific time interval, as illustrated in Figure 2 [53,49]. This allows the model to incorporate temporal dynamics into its training, enhancing its ability to track and analyse transaction sequences over time.

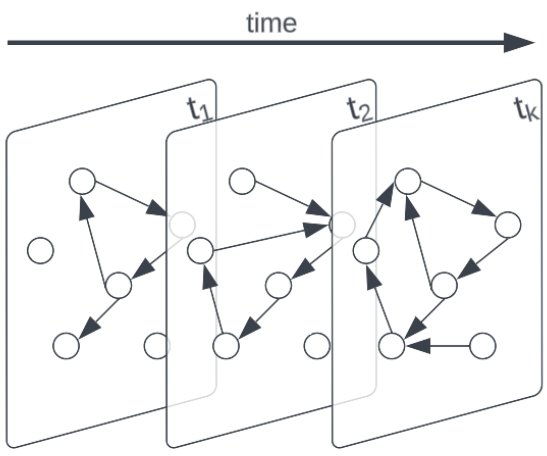


Fig. 2. Temporal structure of the transaction data, where the data at each time slice t form a graph.

2.6 Graph Neural Networks

Graph Neural Networks (GNNs) are machine learning models designed for graph-structured data, capturing the relational and structural information that traditional, tabular models cannot. They are particularly effective for detecting financial crimes, where relationships between entities, accounts and transactions, are crucial. GNNs iteratively aggregate and transform information from a node's neighbours, with the ability to learn both local and global patterns. In financial networks, this means identifying complex transaction patterns indicative of fraud. Various variations of GNNs are considered in this research, the next sections will discuss Graph Convolutional Networks (GCNs), Graph Attention Networks (GATs), Graph Isomorphism Networks (GINs), and Message Passing Neural Networks (MPNNs) shortly.

2.6.1 Graph Convolutional Networks (GCN) Graph Convolutional Networks (GCNs) excel at capturing relational structures within transaction data. They have the ability to model the complex network of relationships between entities such as account holders, merchants, and banks. By representing interactions as graphs, GCNs can identify suspicious patterns using the relational structure of transaction data, also improving explainability, trust, and regulatory compliance in transaction monitoring [39]. GCNs aggregate information from neighbouring nodes, allowing the model to learn representations that reflect the local structure of the graph. This capability is essential for detecting fraud, where the local network context can provide critical clues about suspicious activity.

2.6.2 Graph Attention Networks (GAT) Graph Attention Networks (GATs) build on the foundation laid by GCNs by introducing an attention mechanism that weighs the importance of different nodes within the graph. This allows GATs to prioritise the most relevant parts of the graph for predictions, improving the accuracy and robustness of the model [47]. In anti-money laundering (AML) contexts, GATs can highlight critical relationships and nodes indicative of illicit activities, enhancing the detection of suspicious patterns in complex and noisy transaction data. By focussing attention on specific nodes, GATs can better work with unbalanced datasets in financial transaction monitoring, where only very small sections of the transactions are fraudulent.

2.6.3 Graph Isomophism Networks (GIN) Graph Isomorphism Networks (GINs) are another type of GNN designed to capture the structural identity of nodes in a graph. This ability is crucial for distinguishing between different transaction patterns, as GINs can recognise when two graphs are structurally identical (isomorphic) [51]. This allows the model to detect similarities between different instances of financial crime. By comparing the structural features of different transaction graphs, GINs can uncover different patterns that may indicate fraudulent activity.

2.6.4 Message Passing Neural Networks (MPNN) Message Passing Neural Networks (MPNNs) operate by passing messages between nodes to aggregate information from their neighbours, making them highly effective in modelling the flow of transactions [19]. MPNNs can capture the movement of funds through a network, helping to identify money laundering schemes that involve multiple intermediaries. By modelling the sequential flow of transactions, MPNNs can detect patterns that are characteristic of complex financial crimes, providing a robust tool for analysts to track and investigate suspicious activities.

2.7 Explainability

The need for explainability in AML efforts is crucial, particularly given the significant consequences of being wrongly labelled as fraudulent, which can be both time consuming and costly. Previous research emphasises the importance of transparency and interpretability in AML models to improve trust, accountability, and regulatory compliance [31]. This need for explainability is especially relevant in unsupervised learning approaches, where models autonomously identify patterns and anomalies in transaction data without labelled training data. Ensuring explainability in unsupervised models is essential for stakeholders to understand and trust the decisions made by these models.

Although various methods have been proposed to enhance the explainability of GCNs, their application to AML is relatively unexplored. Techniques such as Sensitivity Analysis, Guided Backpropagation, Layerwise Relevance Propagation, and Autograd-based implementations have shown promise in providing insight into model decisions in general contexts [2].

Existing explainability methods offer varying degrees of success in real-world applications. Techniques such as Sensitivity Analysis and Layerwise Relevance Propagation can provide insight into the decision-making process of, e.g., GCNs, allowing analysts to trace back how specific inputs influence outputs [33]. However, these methods may still fail to fully capture the complexity of financial transaction networks, especially when dealing with incomplete and noisy data. More effective techniques are likely those that incorporate domain-specific knowledge and focus on the temporal and relational aspects of the data, such as explainability methods tailored for these temporal models. For example, visualising attention scores in GATs can highlight critical connections and transactions. This should provide a clearer understanding of why certain transactions are flagged as suspicious [47].

Moreover, using models that work with graph networks inherently improves the interpretability of the output. Graphs offer a visual representation of patterns, allowing analysts to show how specific inputs lead to particular outputs. Although this approach does not fully explain the underlying reasons for labelling a transaction as suspicious, it does provide a transparent view of the situations that led to such conclusions.

2.8 Related work at Avanade

This research is written at the company Avanade. Avanade is an IT consultancy that operates in roughly all industries, among which banks and capital markets. Helping their clients transform their businesses through digital innovation through “the power of people” and Microsoft. Avanade has built experience in the field of Anti-Money Laundering for transaction monitoring. Therefore, this section will highlight what is known and provide background information on the problem based on previous research and experience within Avanade. Previous research has focused on stakeholder involvement and what type of ML model suits detecting money laundering, and the experience of Avanadi contributes to knowledge on the current state of data management within banks.

2.8.1 Stakeholder involvement. Internal research concluded that involving stakeholders throughout the ML development process is essential to ensure that the ML model is aligned with business goals, regulatory requirements, and ethical standards. Stakeholders include representatives from financial institutions, regulatory bodies and collaborative initiatives, such as data scientists, legal and ethical teams, Know Your Customer (KYC)/ Customer Due Diligence (CDD) analysts, and product owners. By getting those stakeholders involved, Avanade learnt a variety of problems and requirements when dealing with AML.
Stakeholders are essential throughout the entire process. Data scientists help assess the feasibility of business requirements and manage expectations. Legal and ethical advisors ensure the model complies with regulations and handles data privacy, security, and fairness. KYC/CDD analysts and regulatory bodies provide insight into the practical needs and regulatory landscape, ensuring the model’s decisions are understandable and actionable.
The internal research concluded that the CRISP-ML(Q) [41] framework should be followed when

approaching an ML project. The CRISP-ML(Q) framework guides the structured and continuous interaction with stakeholders, ensuring regular check-ins and approvals at each phase. This approach helped address challenges like reducing false alerts and adapting to changing criminal behaviour. The stakeholders emphasised the need for interpretable models and detailed explanations of the model's decisions, which are crucial for analysts and regulators.
In summary, the key recommendations for stakeholder inclusion involve:

- *Agile Development Process*: Following the CRISP-ML(Q) cycle, establishing clear gates for stakeholder sign-off and ensuring continuous engagement.
- *Data Scientist Involvement*: From the initial stages, to assess feasibility and manage expectations.
- *Legal and Ethical Consultation*: To ensure compliance with regulatory requirements and fairness in model decisions.

2.8.2 Types of ML models. Similar to the related literature referenced above, the internal research at Avanade looked into the various ML techniques to detect money laundering. Starting with simpler, interpretable models and progressing to more complex ones. Four sets of techniques were evaluated: supervised learning, unsupervised anomaly detection (AD), self-supervised learning (SSL), and semi-supervised approaches using graph data.

Supervised learning models, such as Extreme Gradient Boosting (XGBoost), were used to classify transactions as normal or suspicious based on labelled data. These models can achieve high accuracy when there is a sufficient amount of labelled data available. XGBoost, for example, uses a combination of decision trees to enhance predictive performance. However, the AML datasets are inherently imbalanced, with only a small fraction of transactions being illicit. This imbalance poses a significant challenge as the model can become biased towards the majority class (normal transactions), potentially overlooking the minority class (illicit transactions). To address this, various resampling techniques like Naïve Random Oversampling (RO) and Synthetic Minority Oversampling Technique (SMOTE) were tested, but they only marginally improved model performance.

Unsupervised AD models, including Isolation Forest and k-Nearest Neighbors (kNN), were tried to detect outliers in the data without the need for labelled data. These models try to identify anomalous transactions that deviate from the norm. Isolation Forest, for instance, isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. Transactions that are quickly isolated are considered anomalies. kNN, on the other hand, detects anomalies based on the distance of a transaction to its nearest neighbours. These properties make these models very useful for finding previously unknown money laundering patterns, however their performance is highly dependent on the feature engineering of the dataset. Thus, without domain-specific risk indicators, the internal Avanade research concluded that these models will struggle to distinguish between normal and anomalous transactions effectively.

Semi-supervised learning (SSL) approaches, such as the Value Imputation and Mask Estimator (VIME), were explored to leverage both labelled and unlabelled data. SSL aims to generate representations of the unlabelled data through well-designed pretext tasks. In VIME, for example, the pretext generator creates a corrupted version of the data by masking certain values, and then VIME trains the model to predict these masked values, thus capturing the underlying structure of the data. These learned representations can then be used for tasks such as classifying transactions. By using both labelled and unlabelled data, SSL methods such as VIME can improve the detection of illicit transactions even when labelled data is scarce. However, the internal Avanade research found that the effectiveness of this approach in AML still depends on the quality of the pretext tasks and the ability to generate meaningful feature representations. They conclude that even though the nature of the models is unsupervised, they only extend the detection realm to 'known unknown' cases rather than fully 'unknown' cases.

Then, graph-based methods, particularly Graph Neural Networks (GNNs). Avanade has started exploring their ability to analyse transactional relationships and networks in AML data. GNNs leverage the structure of transaction networks to detect complex money laundering patterns. So, especially money laundering in the layering phase can be tackled here. These models utilise message-passing techniques, where information is propagated through the network to generate node embeddings that capture the local and global structure of the transaction graph. Adaptations like Reverse Message Passing and Port Numbering enhance GNNs' ability to handle the directional nature of financial transactions and distinguish multiple edges between nodes, see Section 3.1. GNNs can thus uncover intricate patterns that traditional methods might miss. However, their application in AML requires substantial computational resources and expertise in graph theory. This is where the current state of research within Avanade ends, the use of leveraging transaction networks to detect money laundering shows promise.

2.8.3 Current access to transaction data within a bank. The previous sections explain why privacy regulations lead to limited access to the full set of transactions. This paragraph will present the transaction data to which a large Dutch bank in the Netherlands has access. The purpose of this section is to give a specific example of what these privacy regulations imply and what data any model within a bank would have access to. This use case and the information used to create it are collected through conversations with (data & AI) consultants at Avanade who work or have worked on transaction monitoring projects at large Dutch banks.

In the context of a large Dutch bank, consider Bank A, which has complete access to its internal transactional data. This includes details such as transaction amounts, timestamps, account numbers, and other relevant metadata. Using various data imputation methods, internal transactions can be completed and enriched without too many limitations imposed by the privacy regulations considered in this research. Data imputation techniques ensure that missing or incomplete data within the bank's internal transactions are filled in using available data.

However, privacy regulations significantly impact the bank's ability to access and utilise data from external transactions, i.e., transactions involving other banks. As shown in Figure 3, transactions from an account at Bank A to accounts at other banks (Bank B or Bank C) and vice versa are common. For these outgoing and incoming transactions, the available data is limited to the amount transferred and the external account name. Detailed information about the external accounts, such as account holder details or the transaction's context, is not available initially due to the privacy constraints. To enrich the information about these external accounts, public databases like the Chamber of Commerce (Kamer van Koophandel (KVK) in Dutch) can be used. These databases can help identify accounts belonging to companies by recognising account names that include designations like Ltd. This enrichment process allows the bank to gather additional context about the external transactions, such as the nature of the business involved, which can be crucial for effective transaction monitoring, as the model can now see one step further than the banks own data.

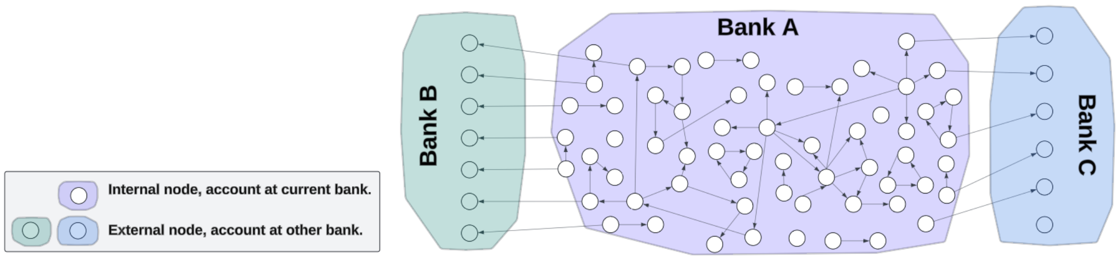

Internal node, account at current bank.
External node, account at other bank.
Bank B
Bank A
Bank C

Fig. 3. Visual representation of the available data for a graph network model at a Large Dutch Bank.

2.9 Problem statement

Dutch banks are currently facing significant challenges in adhering to the Dutch Money Laundering and Terrorist Financing Prevention Act (Wwft). This is largely due to outdated rule-based transaction monitoring systems, which are inadequate in the face of evolving money laundering tactics. The primary issues can be summarised as follows:

1. **Outdated Transaction Monitoring Systems**: The current transaction monitoring systems in large Dutch banks are outdated and rely on static, rule-based models. These models are slow to update and unable to adapt quickly to the changing methods of money launderers. Adjusting or creating new rules within these systems can take several months, which is insufficient given that money launderers change their tactics frequently.
2. **High Rate of False Positives**: The existing rule-based models produce a large number of false positives due to their need to cover all potential true positive cases. This leads to substantial costs as each flagged transaction, whether false or not, requires manual investigation. The manual investigation process is time-consuming and resource-intensive, placing a significant burden on banks.
3. **Hesitancy to Implement Machine Learning Models**: Despite the potential benefits of machine learning models, such as quicker adaptation to new money laundering tactics and reduced false positives, banks are hesitant to implement these solutions. This comes from the fear of large fines and the current lack of thorough exploration and validation of these models in real-world scenarios.
4. **Limited Access to Comprehensive Transaction Data**: Privacy regulations in the Netherlands prevent any single entity, such as Transaction Monitoring Nederland (TMNL), from accessing the complete set of transaction data across all banks. The efforts of TMNL are hindered by these data access limitations. Under current legal frameworks, TMNL must use pseudonymization and cannot access the full scope of transactions, especially those involving private individuals.
5. **Money Launderers Exploiting Multiple Banks**: Money launderers often spread their activities across multiple banks to evade detection. This dispersal of illicit transactions across different financial institutions creates a fragmented view of the money laundering network, making it difficult for any single bank to detect the full scope of illegal activities.

This research aims to explore approaches to work with temporal graph networks and to manage missing information in graph networks for AML transaction monitoring, considering the current limitations and challenges in data access and system adaptability.

3 Algorithms/Methods

To evaluate the added value of incorporating a component specifically focusing on the incomplete networks, this research uses an existing AML classification model. The reason for this is to have an existing and evaluated baseline to be able to focus on the changes in results, not on the overall ability to classify illicit behaviour, and the implications of limited data access. In addition to the incomplete network component, the importance and effect of the temporal component is considered. The following sections dive into the algorithms of the baseline, the temporal components, and the incomplete network component.

3.1 Baseline models: Multi Graph Neural Networks (Multi-GNN)

The Multi-GNN for Anti-Money Laundering project is a repository on GitHub [22], it consists of four Graph Neural Network model classes (GIN, GAT, PNA, RGCN) and three model adaptations for financial crime detection as described by Egressy et al.[19]. The base models are described in their respective papers:

1. **Graph Isomorphism Network (GIN)** by Xu et al. [51]: Is a Message Passing Graph Neural Network based on the Weisfeiler-Lehman graph isomorphism test. This enables differentiation between graphs that are not "isomorphic" to each other (do not have similar structures).

2. **Graph Attention Network (GAT)** by Veličković et al. [47]: Is a Graph Attention Network that uses attention mechanisms to learn the importance of neighbouring nodes' features in a graph. By assigning different weights to different nodes, GATs can focus on the most relevant parts of the graph.

3. **Principal Neighbourhood Aggregation (PNA)** by Corso et al. [13]: Is an architecture which combines multiple aggregators with degree-scalers to capture the diverse aspects of node neighbourhoods. These scalars are functions of the number of messages being aggregated, thus allowing the network to amplify signals based on the degree of each node.

4. **Relational Graph Convolutional Network (RGCN)** by Schlichtkrull et al. [39]: Is a Graph Convolutional Network extended to handle multi-relational data, where edges can have different types or labels. It incorporates relation-specific transformation matrices to model the interactions between different types of relationships in the graph.

Then the paper "Provably Powerful Graph Neural Networks for Directed Multigraphs" by Egressy et al. [19] expands these base models with three adaptations to improve the detection of fraud patterns:

1. **Reverse Message Passing** enhances Message Passing Neural Networks (MPNN) by letting nodes receive messages from outgoing neighbours. This allows counting the outgoing edges and distinguishing node types based on the direction of the edges. Thus, differently from the Message Passing mechanism, the idea is to consider the edges bidirectional however since the directionality is essential in the context of AML to distinguish who sends the transaction to whom, the current state ($h(v)$) and neighbour embedding ($a(v)$) include separate message-passing layers for the in-going and outgoing edges. This is implemented as depicted in Algorithm 1.
   Added value. Reverse MSP allows the model to distinguishing nodes based on the directionality of transactions. In AML, this means the model is able to differentiate between entities that predominantly send money and those that predominantly receive money. This distinction is essential because fraud patterns often involve specific transaction flows, such as money quickly moving through multiple accounts (layering) or being collected into a single account (integration).

Algorithm 1 REVERSE_MSP.

```
(a) Initialize adjacency lists for incoming and outgoing neighbors with timestamps:
    $adj\_list\_in, adj\_list\_out = to\_adj\_nodes\_with\_times(data)$
(b) Aggregate incoming and outgoing messages separately:
        $a_{in}^{(t)}(v) = \text{AGG}_{in}\left(\{h^{(t-1)}(u) | u \in N_{in}(v)\}\right)$
        $a_{out}^{(t)}(v) = \text{AGG}_{out}\left(\{h^{(t-1)}(u) | u \in N_{out}(v)\}\right)$
(c) Update node features: $h^{(t)}(v) = \text{UPDATE}\left(h^{(t-1)}(v), a_{in}^{(t)}(v), a_{out}^{(t)}(v)\right)$
```

2. **Directed Multigraph Port Numbering** assigns local IDs, or port numbers, to each neighbour in a directed multigraph, allowing nodes to distinguish between messages from the same or different neighbours. Egressy states that, since using unique account numbers leads to overfitting, adding port numbers and ordering the neighbours based transaction timestamps should enable nodes to identify repeated messages. This is implemented by looking at the adjacent nodes again, and for each node assigning unique port numbers to its neighbours, sorted by timestamps, see Algorithm 2.

   Added value. Port numbering is crucial for recognizing repeated transactions between the same accounts. It allows the model to track these repeated interactions accurately, identifying sophisticated fraud patterns such as structuring (smurfing), where large amounts of money are broken down into smaller, less suspicious transactions.

Algorithm 2 PORT_NUMBERING.

```
(a) Initialize adjacency lists for incoming and outgoing neighbors with timestamps:
    $adj\_list\_in, adj\_list\_out = to\_adj\_nodes\_with\_times(data)$
(b) Assign port numbers based on sorted timestamps:
    For each node $v$ with neighbours $nbs$:
      – Sort $nbs$ by timestamps.
      – Assign unique port numbers to each neighbour.
(c) Update edge features with port numbers:
      – $ports = ports(edge\_index, adj\_list\_in)$
      – Append $ports$ to the edge features.
```

3. **EGO IDs** assign a unique feature to a central node. This enables the model to detect cycles by recognizing when messages return to the start node. While ego IDs alone struggle with detecting longer cycles, their combination with reverse message passing and port numbering significantly enhances the model's ability to identify cycles, scatter-gather patterns, and bipartite subgraphs (Section 4). This is implemented by assigning unique IDs to each node, allowing the network to distinguish any directed subgraph pattern effectively, see Algorithm 3.

   Added value. Ego IDs are particularly valuable for spotting circular transactions and complex structures within the transaction graph. They help the model identify cycles and patterns where money moves through various accounts and eventually returns to the origin (layering or integration stages).

```
Algorithm 3 EGO_IDs.
(a) Initialize unique IDs for central nodes:
    Assign a distinct binary feature to the ego node.
(b) Combine with reverse MP and port numbering:
    – Use reverse MP to handle edge directions.
    – Apply port numbering to distinguish repeated messages.
(c) Update node features with ego IDs:
    – $h^{(t)}(v) = \text{UPDATE}\left(h^{(t-1)}(v), a_{in}^{(t)}(v), a_{out}^{(t)}(v)\right)$
    – Incorporate ego IDs into the feature update process.
```

Furthermore, the paper by Egressy et al. [19] uses edge updates (EA) as described in Battaglia et al. [8] since AML is a classification problem.

3.1.1 Use and parameters of the baseline.

The described base models and adaptations are extensively experimented in the paper by Egressy et al.[19]. As summarised in Table 1, their experimental setup uses GIN as the main GNN baseline model, with the adaptations reverse message passing, port numbering, and ego IDs added on top (Multi-). PNA and R-GCN are also used as base models, with adaptations added to PNA as Multi-PNA and Multi-PNA+EA.

This research also includes GAT, Multi-GAT, and Multi-GAT+EU as baseline models to complete the comparison.

Baseline results. The paper by Egressy et al. [19] concludes that the adaptations significantly improve the minority class F1 score. For the AML Small HI dataset4.1, the score improves from 28.7% to 57.2% with adaptations, mainly due to reverse message passing and port numbering. Multi-PNA+EU outperforms all baselines on AML datasets, demonstrating the effectiveness and versatility of the approach. Most illicit transactions belonging to money laundering patterns (see Section 4) are identified (layering), although lone illicit transactions (placement) remain challenging to detect.

Table 1. The models from Multi-GNN with extensions and results from Egressy et al.[19] considered in this research. (NR means that for this model there are no results in the paper, however they are considered in this research.)

Model	Paper results
GIN	28.70±1.13
PNA	56.77±2.41
GAT	NR
R-GCN	41.78±0.48
Multi-GIN	57.15±4.99
Multi-GIN+EU	64.79±1.22
Multi-PNA	64.59±3.60
Multi-PNA+EU	68.16±2.65
Multi-GAT	NR
Multi-GAT+EU	NR

3.2 Temporal Components: Time Delta's and Dynamic GAT (DynGAT)

Adding a temporal component can be done in various ways. This research considers two methods; adding time delta's and adding temporal embeddings from a Time Encoder function. Both adaptions are described in the following sections.

3.2.1 Time Delta's The time delta adaption in the Graph Neural Network models is designed to capture the temporal dynamics of the transactions by incorporating the time intervals between consecutive transactions as edge features. By adding these features, the models can detect patterns related to the frequency and timing of transactions. See Algorithm 4, where the steps to calculate the time delta's are explained. Here, the index k is the total number of edges connected to a specific node v, and i is the index of an individual edge within the list of k edges.

Algorithm 4 `COMPUTE_TIME_DELTAS`.

Step	Description	Formula
1	**Initialize Time Deltas:** Initialize a vector of zeros, Δt, with the same length as the number of edges.	$\Delta t_i = 0 \quad \text{for all } i \in E$
2	**Iterate Over Each Node:** For each node v, retrieve the list of k edges $E(v)$ connected to v.	$E(v)$
3	**Sort Edges by Timestamps:** For each node v, sort the k edges $E(v)$ by their timestamps t.	$E(v) = \{(e_1, t_1), (e_2, t_2), \ldots, (e_k, t_k)\}$ such that $t_1 \leq t_2 \leq \ldots \leq t_k$
4	**Calculate Time Deltas:** Compute the time delta for each consecutive pair of k edges.	$\Delta t_{e_i} = \begin{cases} 0 & \text{if } i = 1 \\ t_i - t_{i-1} & \text{for } i > 1 \end{cases}$
5	**Assign Time Deltas:** Assign the calculated time deltas back to the corresponding edges.	$\Delta t(e_i) = \Delta t_{e_i}$ for each edge $e_i \in E(v)$

3.2.2 DynGAT The Dynamic Graph Attention Network (DynGAT) was developed by T. Wei, B. Zeng, W. Guo, Z. Guo, S. Tu, and L. Xu in their 2023 paper, "A Dynamic Graph Convolutional Network for Anti-Money Laundering." This model converts transaction data into a dynamic graph sequence to capture temporal patterns explicitly. Figure 4 shows the original model from the paper [49]. DynGAT uses a Graph Attention Network (GAT) for node embeddings and a Time Encoder (TE) function to incorporate time features, combining these in multi-head self-attention blocks. This Time Encoder is the adaption that is added to the Multi-GNN models. This section first dives into the model as described in the paper and the implementation made available on GitHub [11], then into the integration of the TE adaption into the Multi-GNN models.

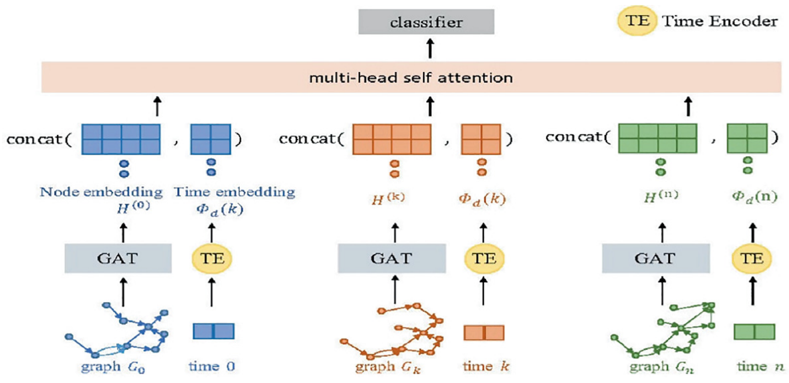


Fig. 4. An overview of the DynGAT model as proposed by Wei et al. [49]

Construction of Graph Sequence from Raw Transaction Records. The DynGAT model works with the Deep Graph Library [44] (DGL) package to implement their graph model. Transaction data is organized into subgraphs over different time intervals, with each subgraph representing transactions within a specific period. Initially, transactions are modeled as static graphs $G = (V, E)$, where vertices $v_i \in V$ represent accounts, and directed edges $e_k = (v_i, v_j) \in E$ represent fund flows, each with a timestamp t_k. Then, the dynamic graph sequence $\mathcal{G} = \{G_0, G_1, \dots, G_n\}$ is formed, with each subgraph defined as:

$$\begin{gathered} G_i = (V, E_i) \\ E_i = \{e_k \mid t^{(i)}_{start} \le t_k \le t^{(i)}_{end}\} \end{gathered} \tag{1}$$

The start and end times for each dynamic graph are calculated as:

$$\begin{aligned} t^{(i)}_{start} &= \frac{i \cdot (T_{max} - T_{min} + 1)}{n} \\ t^{(i)}_{end} &= t^{(i)}_{start} - 1 \end{aligned} \tag{2}$$

Node and Time Embeddings each have their own encoding process before they are concatenated for the model.

- *Node Embeddings via GAT.* For each subgraph, node embeddings are calculated incorporating all the features of the transaction edge. The similarity (s_{ij}) between an account node (v_i) and its transaction counterparties (v_j) is normalised to obtain the attention coefficients (α_{ij}), integrating the edge features (f_{ij}):

  $$s_{ij} = \mathrm{softmax}(\mathrm{LeakyReLU}(\mathbf{a}^T[Wh_i || Wf_{ij} || Wh_j])) \tag{3}$$

  The node embeddings (h_i) are updated as a weighted sum of neighbours' representations (h_j):

  $$h_i = \sum_{j \in N_i} \alpha_{ij} h_j \tag{4}$$

- *Time Embeddings via Time Encoder.* The Time Encoder is designed to encode temporal information into a continuous representation. It used sinusoidal functions to represent periodic time data, so by converting timestamps into a high-dimensional space. Thus, this encoding tries to capture the underlying periodicities and relationships instead of using raw timestamps, to highlight the temporal patterns and anomalies. This could highlight periodic transactions or unusual timing patterns. The Time Encoder class is an attribute of the GAT model class. In the forward function, the time features are encoded using the `self.time_encoder(self.time_cuts)` attribute. The Time Encoder class is depicted in Algorithm 5.
  A time encoder module converts the time sequence into embeddings, which are concatenated with node embeddings:

  $$\overline{h_i}^{(k)} = [h_i || \varPhi_d(n)] \tag{5}$$

Multi-Head Self-Attention Block To capture temporal patterns, the concatenated embeddings are arranged into a set $\left\{\overline{h_i}^{(k)} \mid k \in (0, 1, \dots, n)\right\}$. Self-attention, which models positional information, is applied:

$$F_i = \mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) V \tag{6}$$

Classifier and Training The model uses a Multi-Layer Perceptron (MLP) Classifier for multi-class classification. It consists of three linear layers: two with Leaky ReLU activations to reduce input dimensions, and a final layer with Softmax activation for output probabilities indicating whether a node is suspicious.

Algorithm 5 TIME_ENCODER.

```
class TimeEncoder(nn.Module):
def __init__(self, expand_dim_, factor=5):
    super(TimeEncoder, self).__init__()
    self.expand_dim = expand_dim_
    self.factor = factor
    self.basis_freq = nn.Parameter((torch.from_numpy(1 / 10 **
                           np.linspace(0, 9, expand_dim_))).float())
    self.phase = nn.Parameter(torch.zeros(expand_dim_).float())

def forward(self, ts):
    batch_size = ts.size(0)
    seq_len = ts.size(1)
    ts = ts.view(batch_size, seq_len, 1)
    map_ts = ts * self.basis_freq.view(1, 1, -1)
    harmonic = torch.cos(map_ts)
    return harmonic
```

3.2.3 Comparison Time Delta's and Time Encoding in GNNs.

The time encoding and time delta methods are two different approaches to incorporating temporal dynamics into GNNs. The time encoding method involves using a time encoder module to convert the time sequence into continuous embeddings, which are then concatenated with node embeddings. This method should add a rich temporal representation, capturing complex patterns. It enhances the model's ability to detect subtle, long-term temporal patterns in transaction data.

On the other hand, the time delta method computes the time intervals between consecutive transactions and adds these as edge features. This simpler approach directly captures the frequency and timing of transactions, making it effective in identifying patterns related to rapid transaction sequences or irregular intervals. So, time encoding offers a more nuanced representation and is better suited for detecting complex, long-term trends, while time deltas are particularly useful for highlighting immediate temporal relationships and transaction flow irregularities.

In terms of complexity, time encoding has a more challenging implementation compared to the straightforward calculation and addition of time deltas. The expected performance of these methods is that time encoding is better in scenarios with long-term temporal dynamics, while time deltas would be advantageous for detecting immediate patterns based on transaction timing.

3.3 Incomplete Graph Networks adaption.

Taking missing information into account can be approached in multiple ways. Either by directly finding the missing links and completing the data before applying a classification model, or by letting the model leverage the inherent structure of the graph [28,53].

3.3.1 Deep Network Representation Learning (DNRL) model

A paper describing a method that considers having incomplete information is ”Research on Node classification Algorithm based on Deep Network representation learning under incomplete information” by Jiachiu Jing [23]. This paper describes a semi-supervised deep network representation learning model (DNRL), designed to handle attribute networks with incomplete structural information. It proposes a dynamic strategy that integrates incomplete structural and attribute information. For this, it uses a semi-supervised variational autoencoder to capture high nonlinear features in the network data and learn low-dimensional node representations.

To utilise the strengths of the DNRL model, the incomplete information features of the DNRL model will be included in the embeddings of the Multi-GNN models [19]. This section will dive deeper into the workings of the DNRL model and how this is integrated into the Multi-GNN model.

Data coding

1. **Positive Pointwise Mutual Information (PPMI)** X. The primary task of the DNRL model is to map high-dimensional network nodes to low-dimensional latent spaces. This model uses the Positive Pointwise Mutual Information (PPMI) matrix as input, a dense matrix that captures higher-order proximity between nodes. The PPMI matrix is constructed as follows:

$$\text{PPMI}(w, c) = \max(\log \frac{P(w, c)}{P(w)P(c)}, 0) \tag{7}$$

   where P denotes the co-occurrence probability matrix, where each row (w) and column (c) represent a node in the graph. Thus the PPMI matrix captures the co-occurance probabilities of nodes, reflecting how often nodes appear together within a certain context. This context (P) is generated with the RandomSurfing model, counting the frequency a certain node is the starting point of random walks that end up at another node. Through k-step iterations using a transition matrix T, allowing restarts with a probability of $1 - \beta$:

$$P_k = \beta \cdot P_{k-1}T + (1 - \beta) \cdot P_0 \tag{8}$$

2. **Joint structure-attribute transition matrix** T combines structural and attribute information linearly to provide a comproehensive representation of the network. It is defined as follows:

   - *Structure Transition Matrix*: For a given adjacency matrix $W \in \mathbb{R}^{n \times n}$, where n is the number of nodes, each element $T^W_{i,j}$ represents the normalized probability of transitioning from node i to node j based on their direct link. This normalization ensures that the transition probabilities sum to one across each row, reflecting the likelihood of moving from one node to another within the network's structure.

$$T^W_{i,j} = \frac{W_{ij}}{\sum_{j \in n} W_{ij}} \tag{9}$$

   - *Attribute Similarity Matrix S*: Constructed using cosine similarity between the attribute vectors A of the nodes. Given $A \in \mathbb{R}^{n \times m}$, where m represents the number of attributes, the similarity $S^A_{i,j}$ between nodes i and j is calculated for the top $l = 10$ values to reduce complexity (otherwise 0):

$$S^A_{i,j} = \begin{cases} \text{CosSim}(A) = \frac{A_i A_j^T}{|A_i||A_j|} & \text{if } S^A_{i,j} \in \{\text{top } l \text{ of } S^A_j\} \\ 0 & \text{else} \end{cases} \tag{10}$$

   - *Attribute Transition Matrix* T^A: Similar to the structure transition matrix, the attribute similarity matrix S is normalised to represent the transition probabilities based on attribute similarities:

$$T^A_{i,j} = \frac{s^A_{i,j}}{\sum_{j \in n} s^A_{i,j}} \tag{11}$$

   - *Joint Transition Matrix* T: Balances the structural and attribute information. For each node, if the structural information T^W_i is completely zero (indicating an isolated node), the attribute information T^A_i is used exclusively. Otherwise, a weighted combination of both matrices is applied using a parameter α, which controls the influence of each type of information:

$$T_i = \begin{cases} T^A_i & \text{if } T^W_i \text{ is all zeros} \\ \alpha T^W_i + (1 - \alpha) T^A_i & \text{else} \end{cases} \tag{12}$$

   This approach ensures that isolated nodes that lack structural links are represented on the basis of their attribute similarities, while nodes with structural connections benefit from a combined representation that leverages both structural and attribute data. The parameter α is tuned to balance these contributions, with higher values increasing the weight of structural information.

Feature extraction The DNRL model uses a semi-supervised variational autoencoder (VAE) to extract features from the PPMI matrix X, resulting in low-dimensional feature vectors $Z \in \mathbb{R}^{n \times d}$. Unlike traditional autoencoders, VAEs learn the distribution of the data rather than focussing on individual samples. The VAE model consists of two main components: an encoder and a decoder. The encoder, also known as inference network, generates a variational probability distribution for the hidden variables. Essentially, it estimates how the hidden variables are likely to be distributed. The decoder, also known as the generation network, then use these hidden variables to reconstruct the original data by generating its probability distribution. The main goal of the VAE model is to minimise the reconstruction loss, which measures how accurately the model can recreate the original data, and the Kullback-Leibler (KL) divergence (Equation 13), which measures the difference between the learnt distribution of the hidden variables and a predefined distribution. This combination should ensure that the model both reconstructs the data well and learns a meaningful and smooth representation of the underlying data distribution.

$$KL(p(x) \| q(x)) = \int p(x) \log \frac{p(x)}{q(x)} dx \tag{13}$$

Expected Results Based on Theory According to the DNRL paper by Jiachiu Jing [23], the model is expected to perform better in node classification tasks compared to traditional network representation models, particularly in scenarios with incomplete structural information. By integrating attribute and structural data, the DNRL model should demonstrate improved robustness against sparse network structures while maintaining a higher classification accuracy. The inclusion of attribute information compensates for missing structural links, ensuring that the model effectively captures the underlying network topology and node attributes.

Implementation The DNRL paper only offers the theory; there is no implementation of the method available. Thus, this research has implemented the above-described sections in Python with PyTorch. Taking into account the use of sparse matrices to account for the large size of the datasets.

4 Data exploration from the single-bank view.

Examining the data is crucial to understand the impact of data privacy regulations and the sharing of transaction data among banks and financial institutions. Due to legal and privacy constraints, the use of real transaction data in this research is not feasible. Instead, this research uses a synthetic dataset, specifically the publicly available IT-AML dataset created by Altman et al. (2024) [1]. Although synthetic data is not identical to real data, it offers several research advantages. For example, it avoids the ethical and legal issues associated with accessing real financial data, provides scalability to generate sufficient data to train robust models, and supports benchmarking to establish performance standards for AML models in a controlled environment. The dataset is labelled and the laundering patterns during the layering phase are explicitly modelled, as detailed in Section 4.3. Additionally, the multi-bank nature of the dataset allows for the study of the consequences of a single-bank view on AML efforts.

4.1 Data Deep Dive: IBM Transactions for Anti Money Laundering.

The "HI-Small_Trans.csv" dataset is used to evaluate the impact of considering single-bank data on AML models. HI denotes a higher illicit ratio, indicating a greater presence of laundering activities, and the smallest dataset is selected due to limited computing resources. Table 2 presents the statistics for all datasets in the IT-AML project. Notably, the smaller datasets are independently simulated and are not subsets of the largest dataset. However, they are chronologically simulated to facilitate train, validate, and test splits, adhering to a 60% / 20% / 20% division as recommended by Altman et al. (2024) [1].

Table 2. Statistics IT-AML datasets (Altman et al. [1])

	Small		**Medium**		**Large**	
	HI	**LI**	**HI**	**LI**	**HI**	**LI**
Date Range HI + LI (2022)	Sep 1-10		Sep 1-16		Aug 1 - Nov	
Timespan	10 days		16 days		97 days	
# of Bank Accounts	515K	705K	2077K	2028K	2116K	2064K
# of Transactions	5M	7M	32M	31M	180M	176M
# of Laundering Transactions	5.1K	4.0K	35K	16K	223K	100K
Laundering Rate (1 per N Trans)	981	1942	905	1948	807	1750

The dataset is offered in tabular form, with features representative of real-world financial transactions. The features included in the dataset are detailed in Table 3.

Table 3. Features in HI-Small transactions dataset.

Feature	**Description**
Timestamp	Moment at which the transaction took place.
From Bank	ID of the bank from which the transaction was send.
Account	ID of the account from which the transaction was send.
To Bank	ID of the bank to which the transaction was send.
Account	ID of the account to which the transaction was send.
Amount Received	The amount of money received.
Receiving Currency	The currency in which the money was received.
Amount Paid	The amount of money send.
Payment Currency	The currency in which the money was received.
Payment Format	The type of transfer (e.g., cheque, reinvestment, etc.).
Is Laundering	Binary value indicating money laundering (=1) or not (=0).

In addition to the transaction data, there is a supplementary text file that contains the patterns processed into the transaction data. As depicted in Table 6, this file is organised by money laundering attempts, labelling each transaction within an attempt according to the type of laundering pattern. For example, the scatter-gather pattern depicted in Figure 5 illustrates how an account $80C93DF00$ from bank 0213 distributes money to multiple accounts, which then consolidate the funds into a single account $8053B01E0$ at bank 0013078. These accounts are highlighted in Figure 6, demonstrating the detailed modelling of laundering behaviours.

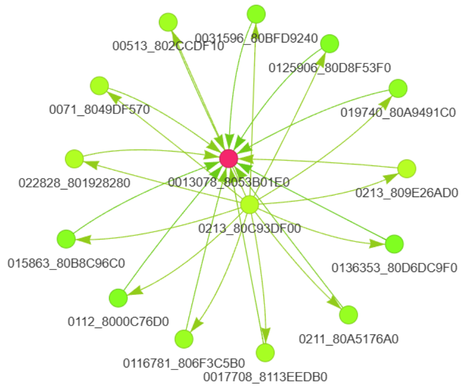
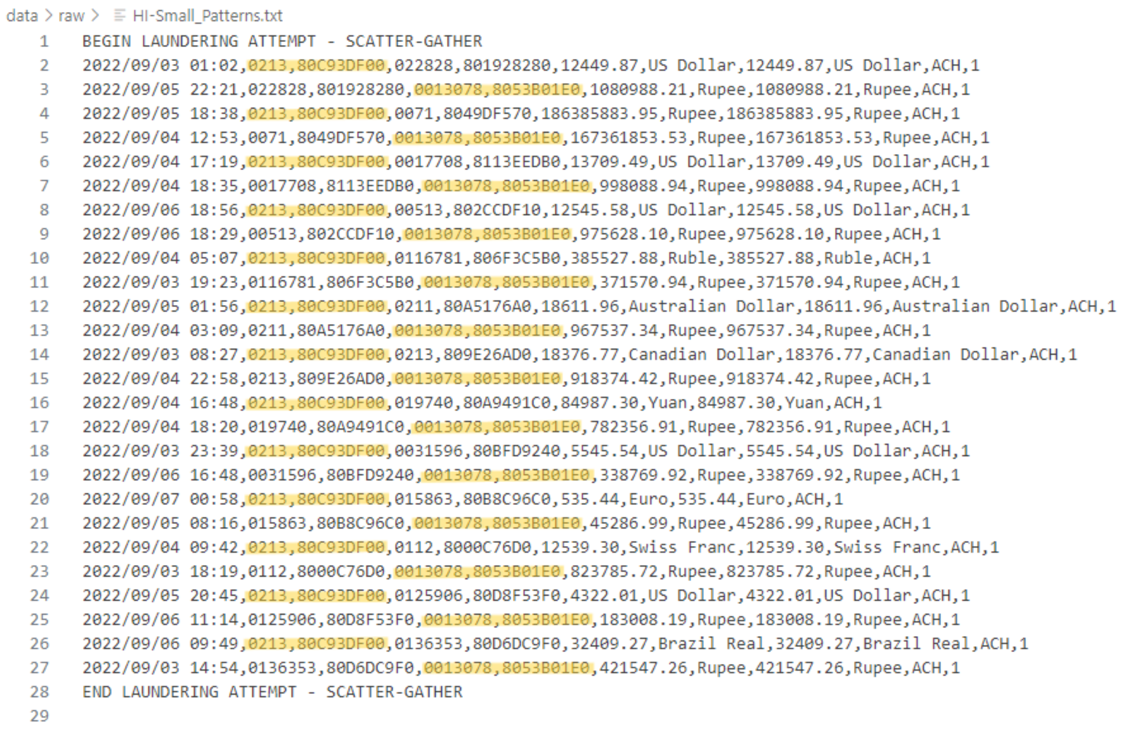


Fig. 5. Scatter-Gather pattern.

The following sections will provide an in-depth analysis of data exploration and preparation. They will also discuss the implications of limited access to the full dataset due to legal and privacy regulations on the recognition of patterns in financial transactions.



```
data > raw > HI-Small_Patterns.txt
BEGIN LAUNDERING ATTEMPT - SCATTER-GATHER
2022/09/03 01:02,0213,80C93DF00,022828,801928280,12449.87,US Dollar,12449.87,US Dollar,ACH,1
2022/09/05 22:21,022828,801928280,0013078,8053B01E0,1080988.21,Rupee,1080988.21,Rupee,ACH,1
2022/09/05 18:38,0213,80C93DF00,0071,8049DF570,186385883.95,Rupee,186385883.95,Rupee,ACH,1
2022/09/04 12:53,0071,8049DF570,0013078,8053B01E0,167361853.53,Rupee,167361853.53,Rupee,ACH,1
2022/09/04 17:19,0213,80C93DF00,0017708,8113EEDB0,13709.49,US Dollar,13709.49,US Dollar,ACH,1
2022/09/04 18:35,0017708,8113EEDB0,0013078,8053B01E0,998088.94,Rupee,998088.94,Rupee,ACH,1
2022/09/06 18:56,0213,80C93DF00,00513,802CCDF10,12545.58,US Dollar,12545.58,US Dollar,ACH,1
2022/09/06 18:29,00513,802CCDF10,0013078,8053B01E0,975628.10,Rupee,975628.10,Rupee,ACH,1
2022/09/04 05:07,0213,80C93DF00,0116781,806F3C5B0,385527.88,Ruble,385527.88,Ruble,ACH,1
2022/09/03 19:23,0116781,806F3C5B0,0013078,8053B01E0,371570.94,Rupee,371570.94,Rupee,ACH,1
2022/09/05 01:56,0213,80C93DF00,0211,80A5176A0,18611.96,Australian Dollar,18611.96,Australian Dollar,ACH,1
2022/09/04 03:09,0211,80A5176A0,0013078,8053B01E0,967537.34,Rupee,967537.34,Rupee,ACH,1
2022/09/03 08:27,0213,80C93DF00,0213,809E26AD0,18376.77,Canadian Dollar,18376.77,Canadian Dollar,ACH,1
2022/09/04 22:58,0213,809E26AD0,0013078,8053B01E0,918374.42,Rupee,918374.42,Rupee,ACH,1
2022/09/04 16:48,0213,80C93DF00,019740,80A9491C0,84987.30,Yuan,84987.30,Yuan,ACH,1
2022/09/04 18:20,019740,80A9491C0,0013078,8053B01E0,782356.91,Rupee,782356.91,Rupee,ACH,1
2022/09/03 23:39,0213,80C93DF00,0031596,80BFD9240,5545.54,US Dollar,5545.54,US Dollar,ACH,1
2022/09/06 16:48,0031596,80BFD9240,0013078,8053B01E0,338769.92,Rupee,338769.92,Rupee,ACH,1
2022/09/07 00:58,0213,80C93DF00,015863,80B8C96C0,535.44,Euro,535.44,Euro,ACH,1
2022/09/05 08:16,015863,80B8C96C0,0013078,8053B01E0,45286.99,Rupee,45286.99,Rupee,ACH,1
2022/09/04 09:42,0213,80C93DF00,0112,8000C76D0,12539.30,Swiss Franc,12539.30,Swiss Franc,ACH,1
2022/09/03 18:19,0112,8000C76D0,0013078,8053B01E0,823785.72,Rupee,823785.72,Rupee,ACH,1
2022/09/05 20:45,0213,80C93DF00,0125906,80D8F53F0,4322.01,US Dollar,4322.01,US Dollar,ACH,1
2022/09/06 11:14,0125906,80D8F53F0,0013078,8053B01E0,183008.19,Rupee,183008.19,Rupee,ACH,1
2022/09/06 09:49,0213,80C93DF00,0136353,80D6DC9F0,32409.27,Brazil Real,32409.27,Brazil Real,ACH,1
2022/09/03 14:54,0136353,80D6DC9F0,0013078,8053B01E0,421547.26,Rupee,421547.26,Rupee,ACH,1
END LAUNDERING ATTEMPT - SCATTER-GATHER
```

Fig. 6. Scatter-Gather pattern from HI-Small_Patterns.txt file.

4.2 Data Understanding

The *HI-Small transactions* dataset includes 515,088 nodes (holdings/ accounts engaging in transactions) and a total of 5,078,345 transactions. The illicit ratio is calculated as 0.10%, derived from 5,177 illicit transactions out of a total of 5,078,345 transactions. The exact division of the dataset into training, validation, and test sets can be found in Table 4.

Table 4. Training, validation, and testing split of the dataset, with respective illicit ratios.

	% of full dataset	**% illicit in subset**
Training set	63.98%	0.08%
Validation set	19.01%	0.11%
Testing set	17.01%	0.19%

Table 5 shows statistics on the transaction amounts in USD divided by transactions labelled as money laundering and not money laundering. Key observations from this table include:

– The **mean** amount received for money laundering transactions is 36,135,310.41, significantly higher than the mean for non-money laundering transactions at 5,957,962.48.
– **Median** amounts also show a substantial difference, with money laundering transactions having a median of 8,667.21 compared to 1,407.51 for non money laundering transactions.
– **Standard deviations** are notably high, indicating significant variability in transaction amounts, especially for non-money laundering transactions.
– **Maximum** amounts received and paid in both categories highlight extreme outliers, with maximum values reaching over 84 million for money laundering and over 1,046 billion for non-money laundering transactions.

These statistics suggest that money laundering transactions tend to involve much larger sums of money compared to regular transactions.

The distribution of currencies shows a predominance of transactions in US Dollars (37%), followed by Euros (23%), with a variety of other currencies including Swiss Francs, Yuan, Shekels, Rupees, UK Pounds, Yen, Rubles, Bitcoin, Canadian Dollars, Australian Dollars, Mexican Pesos, Saudi Riyals, and Brazilian Reals each having smaller shares. The payment formats also vary, with cheques (37%) being the most common, followed by credit cards (26%), ACH[2] (12%), cash (10%), reinvestment (9%), wire transfers (3%), and Bitcoin (3%).

Table 5. Statistics on the transaction amounts.

Statistic	**Amount Received**		**Amount Paid**	
	Money Laundering	**No Money Laundering**	**Money Laundering**	**No Money Laundering**
Count	5,177	5,073,168	5,177	5,073,168
Mean	36,135,310.41	5,957,962.48	36,135,310.41	4,477,000.04
Median	8,667.21	1,407.51	8,667.21	1,410.99
Std.Dev.	1,527,918,669.80	1,036,563,448.52	1,527,918,669.80	868,846,296.80
Min	0.003227	0.000001	0.003227	0.000001
25%	2,634.97	183.07	2,634.97	184.16
50%	8,667.21	1,407.51	8,667.21	1,410.99
75%	18,832.27	12,322.51	18,832.27	12,279.34
Max	84,853,144,179.58	1,046,302,363,293.48	84,853,144,179.58	1,046,302,363,293.48

[2] An ACH (Automated Clearing House) transfer is a low-cost, electronic transaction system used in the United States for direct deposits, bill payments, and transfers, typically taking 1-3 days to process.

4.2.1 Banks The dataset includes a total of 30, 528 sending banks and 15, 850 receiving banks. Notably, all the receiving banks are also sending banks, which implies that there are 14, 678 banks that only send transactions but do not receive any. Among all these banks, 99 receiving banks have transactions that are part of a laundering pattern, and 100 sending banks have transactions associated with laundering. Intercepting at 27 banks which are involved in both sending and receiving transactions that are part of laundering patterns.

The top 3 banks in terms of total transaction amounts sent and received show significant financial activity, see Table 6. The amount sent by the top bank is 6.6% of the incomprehensible 22.9 trillion that is sent in total in this dataset. Similarly, the amount received by the top bank is 6.3% of the total amount received.

Table 6. Top 10 banks in terms of total transaction amounts sent and received.

Bank ID	Amount sent	Bank ID	Amount received
022164	1, 509.86 billion	0116425	1, 915.59 billion
0112064	1, 165.08 billion	022164	1, 489.89 billion
019925	1, 123.37 billion	0210185	1, 437.73 billion

Additionally, the transaction frequency is shown in Table 7. This shows that Bank ID 070 is the most active sender with 449, 859 transactions, while Bank ID 012 is the most active receiver with 34, 732 transactions. These are big players in the transaction network, however, when looking at the illicit behaviour in Bank 070 there are no transactions sent from this bank that are part of transaction patterns, solely transactions labelled as fraud due to placement and integration. Also interesting to note, that 3, 449 banks only send money once, and 3, 854 banks only received money once, meaning they occur only once in the entire dataset.

The bar graphs in Figure 7 show the average transaction amounts paid and received by banks, distinguishing between money laundering and non-money laundering transactions.

- The top left graph shows the **mean amounts paid** by banks in **non-money laundering** transactions, with Bank ID 0136334 having the highest mean payment, significantly higher than other banks at 434.16 million.
- The top right graph illustrates the **mean amounts paid** in **money laundering** transactions, where Bank ID 0014384 stands out with an exceptionally high mean amount of 20 billion.
- The bottom left graph presents the **mean amounts received** by banks in **non-money laundering** transactions, with Bank ID 0116425 receiving the highest average amount of 330.45 million.
- The bottom right graph indicates the **mean amounts received** in **money laundering** transactions, again highlighting Bank ID 0116781 with the highest average of 20 billion, followed by Bank ID 0042596 with 15 billion.

Table 7. Transactions Sent and Received by Banks and Accounts

Transactions Sent from Bank #			Transactions Received from Bank #		
ID	# of Transactions	% from Total	ID	# of Transactions	% from Total
070	449,859	8.8584%	012	34,732	0.6839%
012	71,177	1.4016%	001	30,115	0.5930%
010	64,791	1.2758%	010	29,926	0.5893%
001	62,211	1.2250%	003	25,627	0.5046%
020	41,008	0.8075%	007	23,029	0.4535%
003	38,413	0.7564%	015	22,730	0.4476%
007	31,086	0.6121%	020	22,048	0.4342%
0211	30,451	0.5996%	028	21,160	0.4167%
0116	30,232	0.5953%	0211	20,576	0.4052%
015	30,027	0.5913%	0116	20,240	0.3986%
Transactions Sent from Account #			**Transactions Received from Account #**		
ID	# of Transactions	% from Total	ID	# of Transactions	% from Total
100428660	168,672	3.3214%	100428660	1,084	0.0213%
1004286A8	103,018	2.0286%	1004286A8	653	0.0129%
100428978	20,497	0.4036%	80F47A310	159	0.0031%
1004286F0	18,663	0.3675%	100428978	150	0.0030%
100428780	17,264	0.3400%	8018859B0	144	0.0028%
1004289C0	16,794	0.3307%	1004289C0	132	0.0026%
100428810	16,426	0.3235%	100428780	117	0.0023%
1004287C8	14,174	0.2791%	100428810	114	0.0022%
100428738	13,756	0.2709%	80F0EF460	109	0.0021%
100428A51	13,073	0.2574%	1004286F0	108	0.0021%

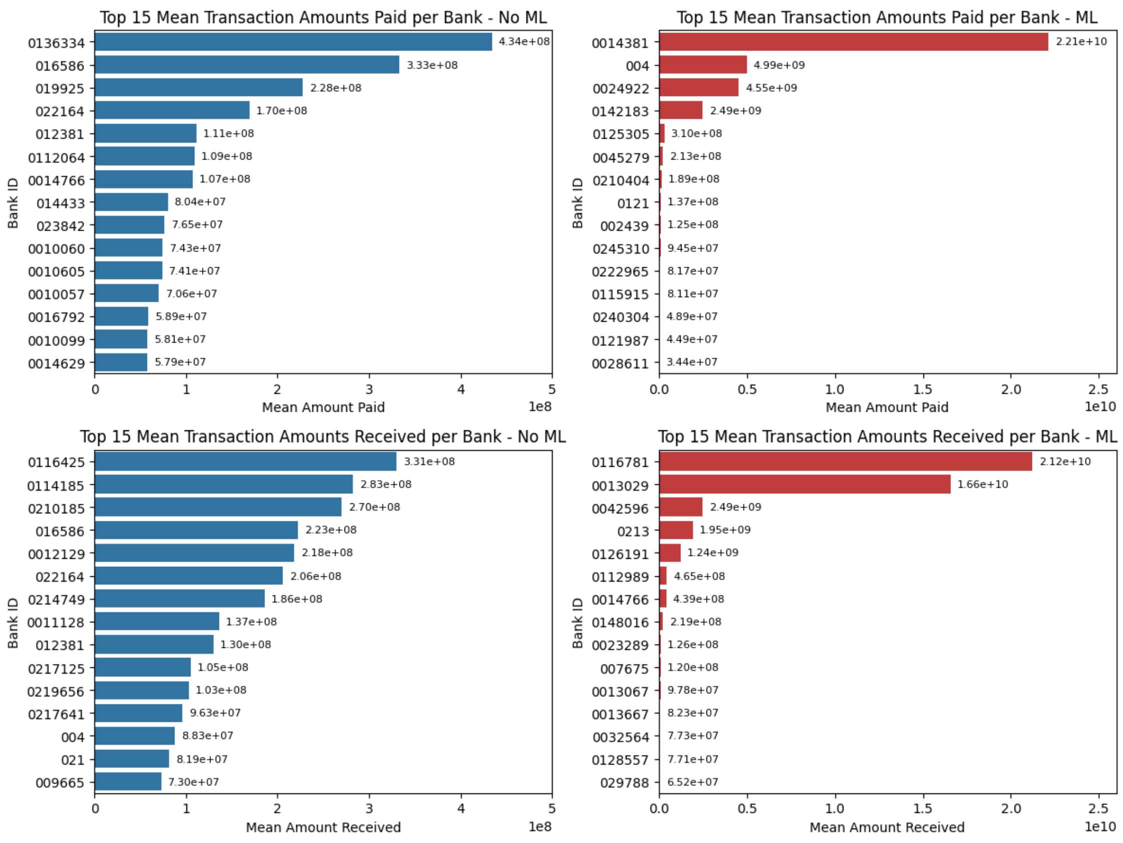


Fig. 7. Bar graphs with the top 15 average transaction amounts paid and received by banks.

4.2.2 Accounts

Together in all the banks, there are 496,995 accounts that send money and 420,636 accounts that receive money. Among these, 402,551 accounts both send and receive money. 114 accounts that send money are part of a laundering pattern, while 110 accounts that receive money are involved in laundering patterns. There are 11 accounts that both send and receive money that is part of laundering patterns.

In terms of total transaction amounts, the highest total amount *paid* by an account is from Account ID 800$E8B7F$0, amounting to 1,505,695,821,886.76, representing 6.58% of the total amount sent across all accounts. The highest total amount *received* by a single account is also from Account ID 800$E8B7F$0, amounting to 1,486,635,895,502.05, which is 4.89% of the total amount received across all accounts. This account is notably active, both in terms of sending and receiving large sums of money.

Figure 8, illustrates the average transaction amounts paid and received by accounts, differentiating between money laundering (ML) and non-money laundering (No ML) transactions.

- The top left graph shows the **mean amount paid** by accounts in **non-money laundering** transactions. Account ID 803D95360 has the highest mean payment of $5.31e+11$, followed by Account ID 804$C1CD$70 with $3.22e+11$.
- The top right graph illustrates the **mean amount paid** in **money laundering** transactions, where the top two to three accounts stand out; Account ID 806$B9AF$90 with a mean payment of $8.49e+10$, followed by Account ID 805$C2AFB$0 with $6.64e+10$.
- The bottom left graph presents the **mean amount received** by accounts in **non-money laundering** transactions. In contrast to the other plots, these accounts have an even spread of transaction amounts. Account ID 8060CEE00 leads slightly with $5.50e+10$, followed by Account ID 8061A4230 with $4.47e+10$.
- The bottom right graph indicates the **mean amount received** in **money laundering** transactions. Similar to the top right, there are two to three accounts that receive by far the most illicit transactions; Account ID 806$B9B6A$0 with $8.49e+10$, followed by Account ID 805$C2B8A$0 with $6.64e+10$.

The frequency of transactions is ranked in the lower half of Table 7. Account ID 100428660 both sent the most transactions, totalling 168,672, and received the most transactions, totaling 1,084.

Transaction variability is measured by the standard deviation of the transaction amounts. Account ID 803D95360 exhibits the highest variability in transaction amounts sent, with a standard deviation of 729,226,853,719. For received transactions, Account ID 8060CEE00 shows the highest variability, with a standard deviation of 166,409,535,930. High variability can indicate inconsistent transaction sizes, which can be a red flag for financial monitoring [10].

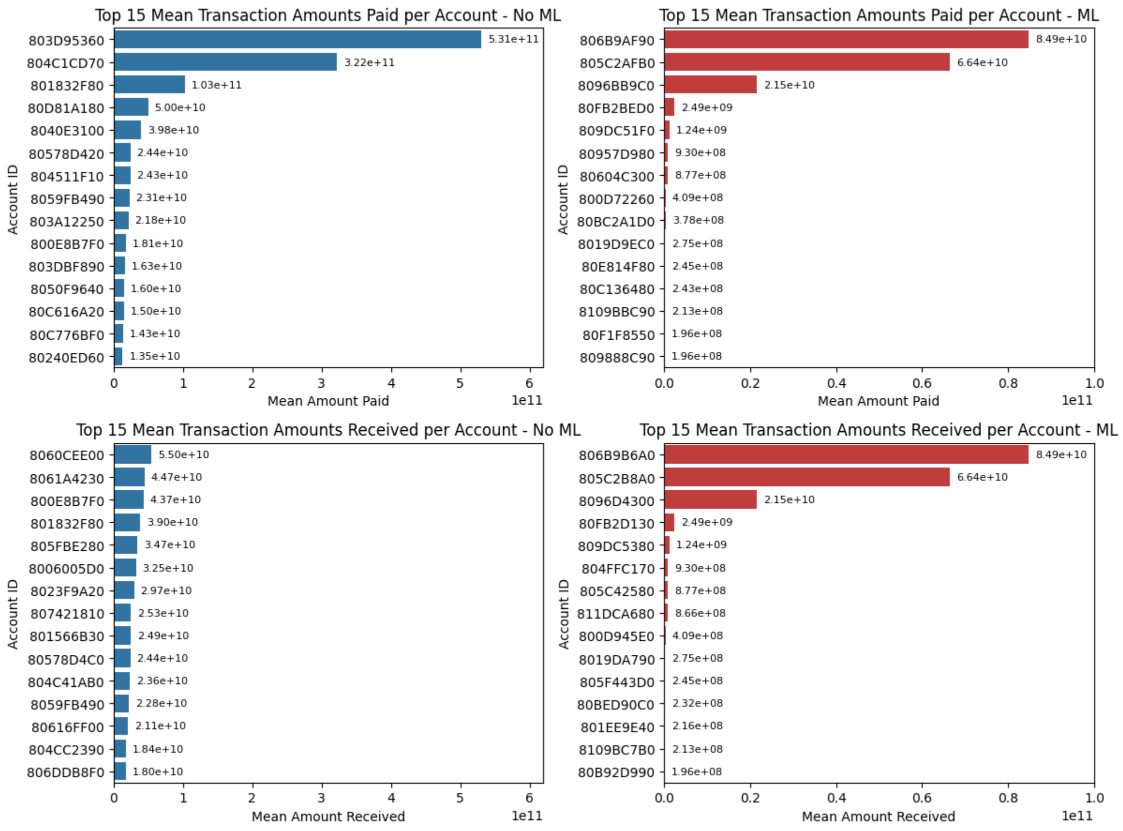


Fig. 8. Bar graphs with the top 15 average transaction amounts paid and received by accounts.

4.2.3 Transactions Diving deeper into the transaction amounts received and paid, the distributions are split by money laundering status, currencies, and payment formats.

Figure 9, illustrates the distribution of transaction sizes received and paid, distinguishing between money laundering and non-money laundering activities. The plots show significant outliers for non-money laundering transactions, where the amounts can reach up to $1 trillion. For both received and paid amounts, Table 5 showed that all statistics, except the maximum, are higher for money laundering transactions. This suggests that transactions flagged for money laundering typically involve larger sums of money. The exclusion to this are the extreme outliers; these are only present in the transactions labelled as not money laundering.

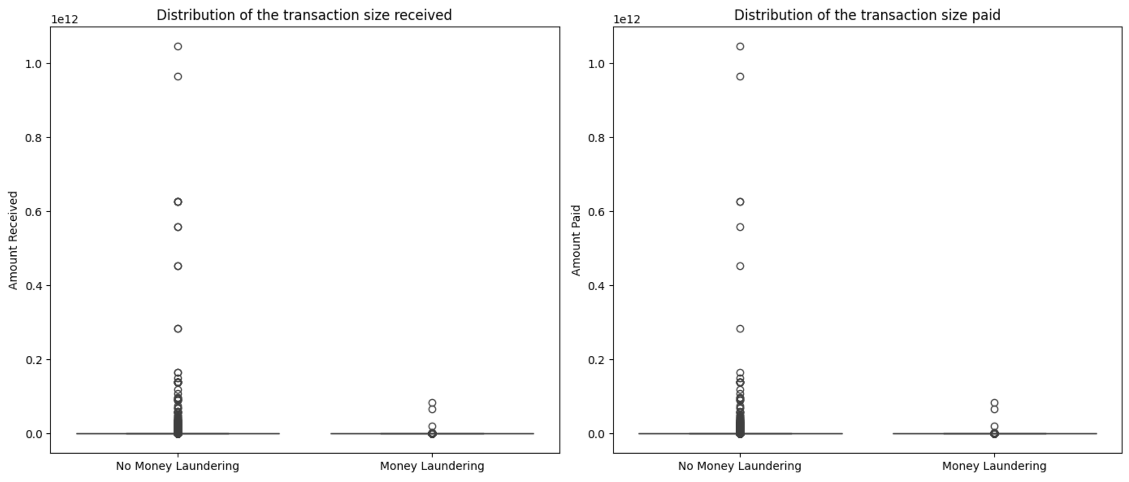


Fig. 9. Boxplots of the destribution of transaction sizes for the Amount Received and Paid, split by Money Laundering status.

Amount Received and Paid split by Currency. The top two boxplots in Figure 10 categorise the amounts received and paid by the different currencies. The Rupee, Yen and Ruble are prominent currencies in these transactions, with substantial variances in the amounts transferred. They include several outliers with large amounts, highlighting the potential for high-value transactions in these currencies. This variability and the presence of outliers in certain currencies indicate the need for vigilant monitoring, as they could be indicative of illicit financial flows.

Amount Received and Paid split by Payment Format. The bottom set of boxplots in Figure 10 break down the transaction amounts by payment format. Cheques and ACH transactions show a considerable range of amounts with notable outliers, suggesting that these formats are commonly used for both small and large transactions. On the other end, Bitcoin transactions have very low values with up to six decimals small. For this reason this entire category of outliers could be excluded, however removing them would reduce the money laundering instances in the dataset too much.

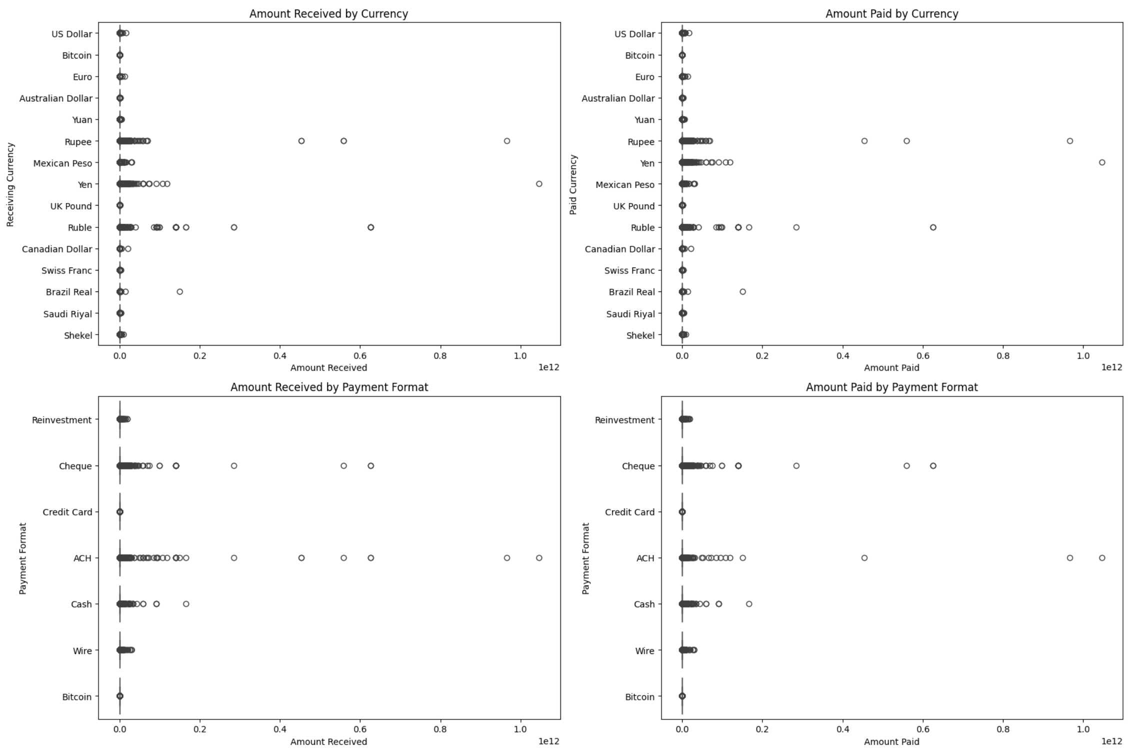


Fig. 10. Boxplots of the destribution of transaction sizes for the Amount Received and Paid, split by the Currencies and by the Payment Formats.

4.3 Laundering patterns of the layering phase.

During the layering phase, illicit funds are moved and disguised through multiple transactions and accounts, to making detection and tracking challenging. Suzumura and Kanezashi[42], identified and illustrated eight common money laundering patterns used in this phase. These patterns should demonstrate the methods launderers use to conceal financial trails and are the patterns simulated into the IT-AML dataset, see Figure 11.

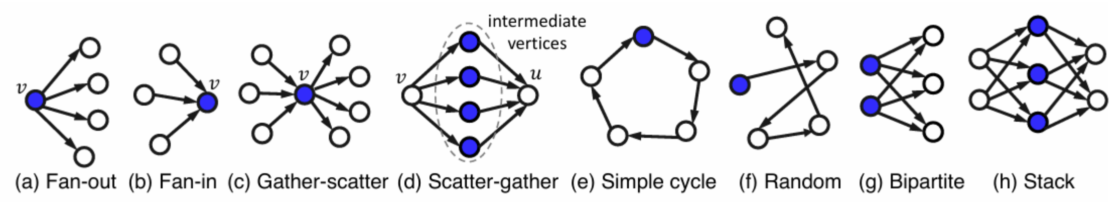


Fig. 11. Laundering Patterns Simulated into IT-AML (Suzumura & Kanezashi, 2021 [42]).

Fan-Out Pattern is a single account distributiong funds to multiple accounts. This is characterized by a vertex v having outgoing edges that connect it to at least $k \geq 2$ different vertices. The goal is to obscure the origin of funds by splitting them into smaller amounts and distributing them across various accounts.

Fan-In Pattern is the reverse of the fan-out pattern. Here, multiple accounts funnel money into a single account. This is illustrated by a vertex v having incoming edges from at least $k \geq 2$ different vertices. The goal is to combine illicit funds into a single account.

Gather-Scatter Pattern combines fan-in and fan-out patterns. It involves a single vertex that first gathers funds from multiple sources and then scatters these funds to multiple destinations. The goal is to enhances the difficulty of tracking the flow of money.

Scatter-Gather Pattern is the opposite of the gather-scatter pattern. Funds are dispersed from one account to several intermediate accounts and then funneled into a single account. This pattern involves a fan-out from one vertex and a fan-in to another vertex, with the same set of intermediate vertices being used.

Simple Cycle Pattern involves a sequence of transactions that form a closed loop, starting and ending at the same account. This pattern can be used to circularly move funds through a series of accounts back to the original owner. The goal is to create a complex trail that is hard to follow.

Random Pattern transfers funds randomly among various accounts, without returning to the original account. This can be viewed as a random walk through accounts owned or controlled by the laundering entity, making the tracking of funds more unpredictable. Also making the pattern itself only defined by randomness and difficult to recognize.

Bipartite Pattern moves funds from a set of input accounts to a set of output accounts. This pattern is used to transfer funds between two distinct groups of accounts, further complicating the detection of money laundering activities.

Stack Pattern builds on the bipartite pattern by adding another layer of transactions, essentially creating multiple bipartite layers. This additional complexity makes it even harder to trace the origin and final destination of the laundered funds.

4.3.1 Patterns in the dataset Each type of pattern is simulated into the transaction data. When solely looking at the transactions that are part of a pattern, the data can be graphed as shown in Figure 12. This visualization was made with the PyVis [18], a library for visualizing interactive graph networks. The full set of transactions labelled as money laundering also include 5, 055 transactions that are not part of a pattern. These are transactions that are labelled as fraud due to placement or integration, not the layering phase.

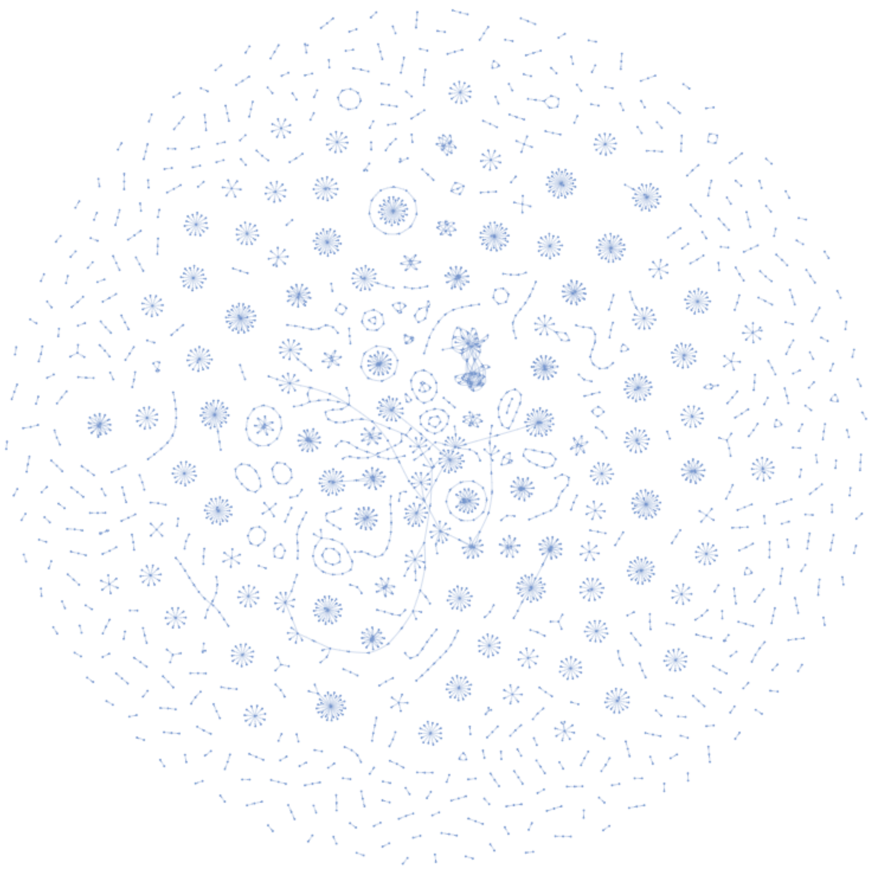


Fig. 12. Pyvis graph of all transactions in a money laundering pattern in the IT-AML dataset.

To ensure that the patterns are properly simulated in the data, a separate graph is plotted for each laundering pattern. See Figures 13, 14, and 15. It is visible that most patterns are generated correctly, and it also includes various sizes of the patterns. The visible exceptions are the Stack, Bipartite, and Random patterns, in accordance with a discussion page on the Kaggle project of the dataset[25], it can be concluded that they are not generated exactly as described by Altman et al. [1]. The stack pattern seems to be only partially generated. Where single transactions and two consecutive transactions are generated, but not the described pattern where multiple accounts each transfer to multiple other accounts and these each send transfers through to another multiple set of accounts. Another thing to note are the smallest groups of nodes in the patterns. Regardles of the pattern, some only have two nodes connected by an edge. These patterns are assumed to be "in progress" laundering. As in real life, where it is not possible to have all patterns complete at one time.

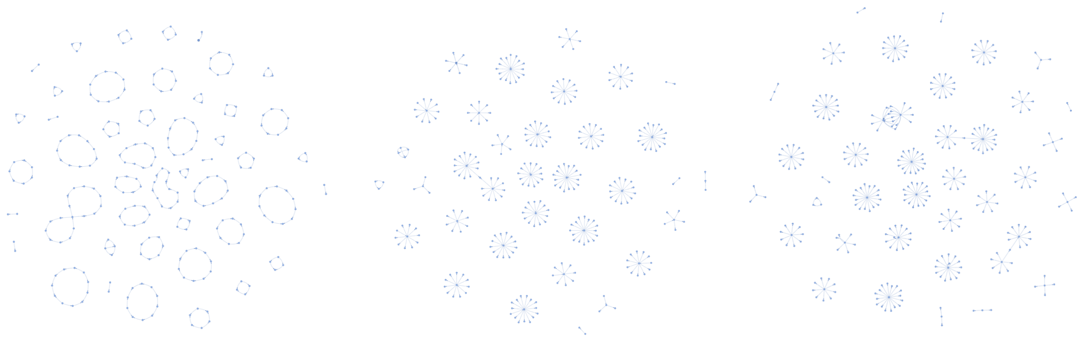


Fig. 13. Simple cycle, Fan-in, and Fan-out patterns.

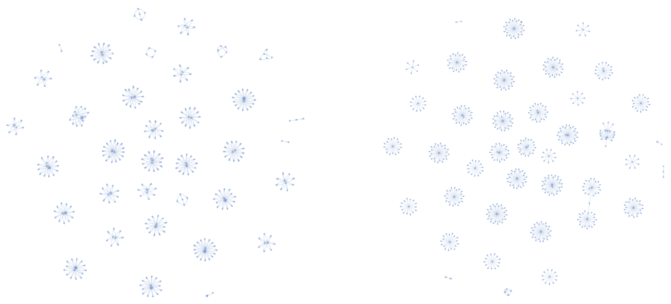


Fig. 14. Scatter-Gather and Gather-Scatter patterns.

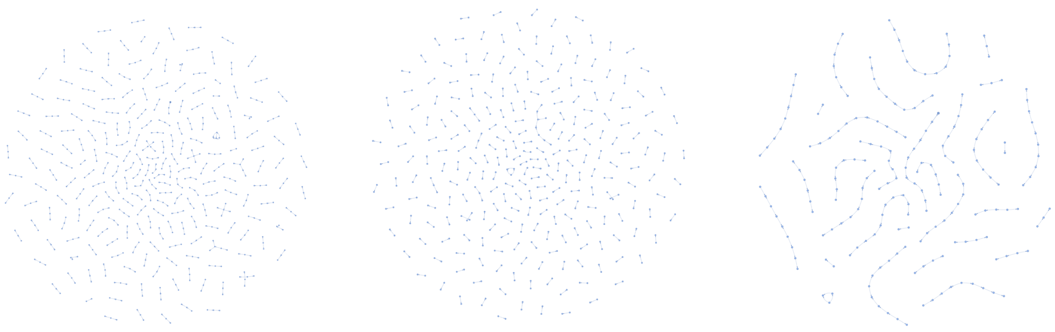


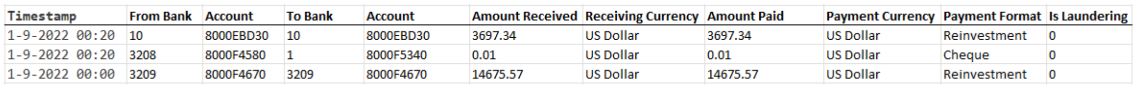
Fig. 15. Stack, Bipartite, and Random patterns.

4.4 Data Preparation

Based on the insights of the previous section, the data preparation was executed. The required changes are limited due to the synthetic nature of the dataset. So does the dataset contain no missing values and limited extreme outliers. Only values greater than $8e11$, as visible in Figure 9, were removed. Besides this, the transactions labelled as a Stack or Bipartite patterns were removed as explained in Section 4.3.1.

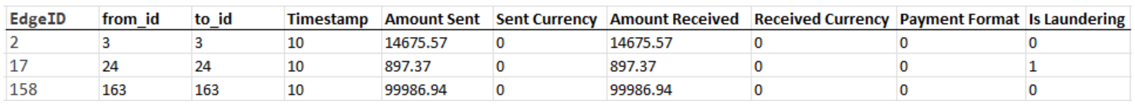
To fit the Multi_GNN models by Egressy et al.[19], this research preprocesses the IT-AML dataset the same way. Initially, the categorical fields in the loaded csv input file, see Figure 16, are converted to integers, such as currencies and payment formats. The transaction timestamp is normalised relative to the first transaction, set to zero. Unique account identifiers are generated by combining bank and account IDs. The data is written in an output file with a clear structure, as depicted in Figure 17. Finally, the output data are sorted by timestamp to organise it chronologically.

This results in a dataset with $5,077,614$ transactions, 8 predictive features, and a target feature that classifies the transaction as fraudulent or not. The labels now indicate that there are $4,448$ transactions related to money laundering which is 0.09% of the full dataset. With fraudulent transactions comprising such a small part of the dataset the problem of a highly unbalanced dataset, as explained in Section 1, is confirmed. A highly unbalanced dataset is problematic because most algorithms will learn the majority class and thus not the fraudulent minority class.



Timestamp	From Bank	Account	To Bank	Account	Amount Received	Receiving Currency	Amount Paid	Payment Currency	Payment Format	Is Laundering
1-9-2022 00:20	10	8000EBD30	10	8000EBD30	3697.34	US Dollar	3697.34	US Dollar	Reinvestment	0
1-9-2022 00:20	3208	8000F4580	1	8000F5340	0.01	US Dollar	0.01	US Dollar	Cheque	0
1-9-2022 00:00	3209	8000F4670	3209	8000F4670	14675.57	US Dollar	14675.57	US Dollar	Reinvestment	0

Fig. 16. Sample of the datset before preprocessing.



EdgeID	from_id	to_id	Timestamp	Amount Sent	Sent Currency	Amount Received	Received Currency	Payment Format	Is Laundering
2	3	3	10	14675.57	0	14675.57	0	0	0
17	24	24	10	897.37	0	897.37	0	0	1
158	163	163	10	99986.94	0	99986.94	0	0	0

Fig. 17. Sample of the datset after preprocessing.

The data splitting method used by Egressy et al.[19] was followed in this research. The transactions are first ordered by their timestamps, which are normalised to start from zero for ease of analysis. The data set is then split into training, validation and test sets based on a temporal 60-20-20 split, following the method used by Egressy et al.[19]. Importantly, the splitting is dynamic, meaning the validation set has access to the previous training data, and the test set has access to the previous validation data, allowing for accurate construction of transaction graphs over time.

Once the dataset is split, it is converted into PyTorch Geometric (PyG[16]) Data objects. This includes initialising node features with placeholder values and selecting relevant edge features such as timestamp, amount received, and payment format. Special attention is paid to calculating the illicit transaction ratios and ensure an even distribution across the splits. Further processing steps are performed, such as the ability to add ports, time deltas, and time encodings to the embeddings, and the features are normalised to prepare the data for model training and evaluation.

4.5 Implications of single-bank view.

This section explores the implications of analysing financial transactions from the perspective of a single bank, providing a detailed examination of how data visibility impacts the detection of money laundering patterns. By filtering the dataset to isolate transactions only visible to individual banks, this section aims to visualise the partial patterns that emerge when only one bank's data is available. By dissecting the dataset, the actual limitations of monitoring transactions without access to the full picture are brought to light. Furthermore, the Multi-GNN models will be trained and evaluated on these single-bank datasets to assess their performance in identifying suspicious activities. These last results can be found in Section 6.2.

4.5.1 Bank selection for single-bank view. To ensure a meaningful and representative comparison, the selection of banks for the single-bank view was based on specific criteria that reflect the overall dataset's characteristics. This means banks with the most occurrences the patterns and a percentage of laundering patterns close to the overall average of 0.1%.

Table 8 presents the top ten banks ranked by occurrences in the patterns. Among these, bank 012 and bank 001 emerged as the most suitable candidates based on the percentage of laundering patterns in the subset of transaction data to which the banks would have access. Bank 012 had the highest occurrences (131) in laundering patterns with a slightly higher percentage than the target at 0.12%, while bank 001 had a lower number of occurrences (89) but exactly matched the target percentage at 0.1%. Interestingly, while bank 012 has the most occurrences in patterns, the set of transactions related to bank 012 only comprises 1.9584% of the full dataset. Similarly, bank 001 only comprises 1.6889% of the full dataset.

Table 8. Single-bank view selection criteria.

Bank ID	**Occurences in patterns**	**% patterns in sub-dataset**
012	131	0.12%
0119	125	0.41%
011	109	0.25%
020	105	0.17%
001	89	0.1%
0222	76	0.58%
023	66	0.17%
0048309	61	0.86%
010	59	0.07%

4.5.2 Visualization of the problem. To visualise the consequences of the single-bank view, the nodes of banks 012 and 001 were isolated. Continuing to focus on the patterns, each pattern is plotted with the nodes of bank 012 in red, their direct neighbours in blue, and all other nodes in grey. This shows what data would be available to a single bank, as explained in Section 2.8.

For the gather-scatter, fan-in, and cycle patterns, the effect is shown in Figures 18, 19, and 20. The scatter-gather, fan-out, and random graphs can be found in Appendix 8.

The Gather-Scatter pattern graph (Figure 18) shows that when highlighting the nodes accessible to bank 012, only two patterns remain complete. Besides these, most patterns are completely lost and six partial gather-scatter patterns remain where the gather-scatter structure is no longer evident. This can significantly hinder the detection of laundering as the partial patterns most likely do not fit the expected criteria.

The Fan-In Pattern graph (Figure 19) similarly shows that only one pattern is complete, most others are fully greyed out, and five partial patterns remain. While some sub-patterns could still suggest a fan-in pattern, those reduced to two or three nodes will be difficult to distinguish by a model. The unrecognizable sub-patterns take up half the pattern clusters thus cannot be ignored as unfinished patterns.

The Cycle Pattern graph (Figure 20) should show transactions forming a closed loop. However, when restricting the view to bank 012 many cycles are broken. This can lead to significant challenges in identifying laundering as the continuity required to recognise cycles is lost.

4.6 Conclusion

Fragmentation and disappearance of transaction patterns due to the single-bank view would significantly impair detection rates. Complex laundering patterns such as gather-scatter, fan-in, and cycles become partial and incomplete, making them harder to recognise. This limited visibility could lead to increased false negatives, where suspicious activities go undetected, and false positives, where innocent transactions are wrongly flagged as suspicious.

The lack of graph connectivity and the presence of only partial subgraphs would make it difficult to identify central nodes or high-flow transaction paths indicative of laundering. Consequently, banks may misclassify transactions.

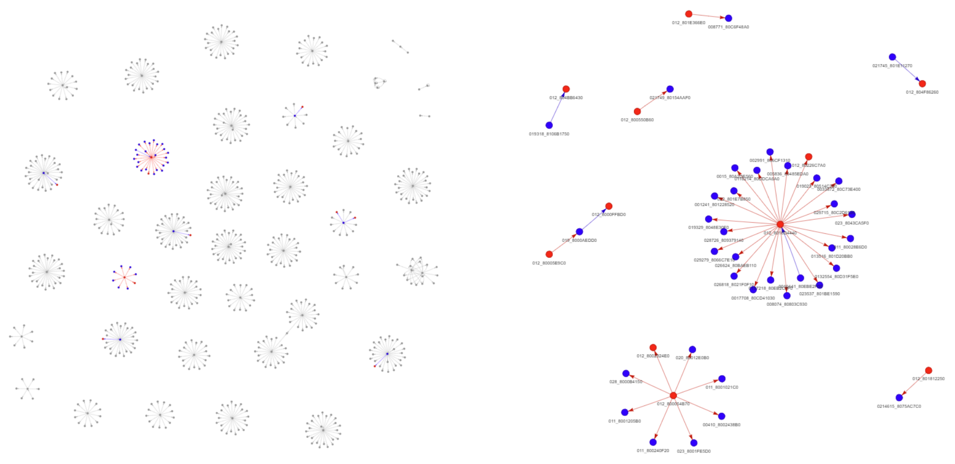


Fig. 18. Gather-Scatter pattern for only bank 012.

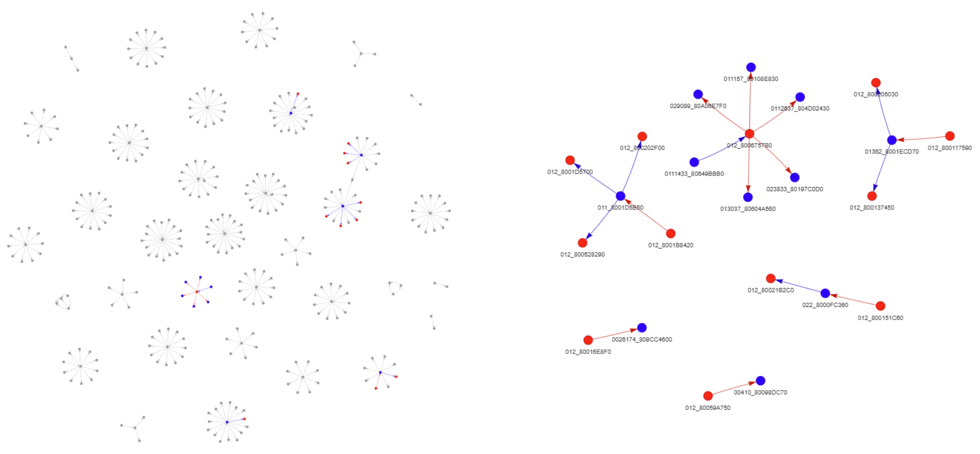


Fig. 19. Fan-in pattern for only bank 012.

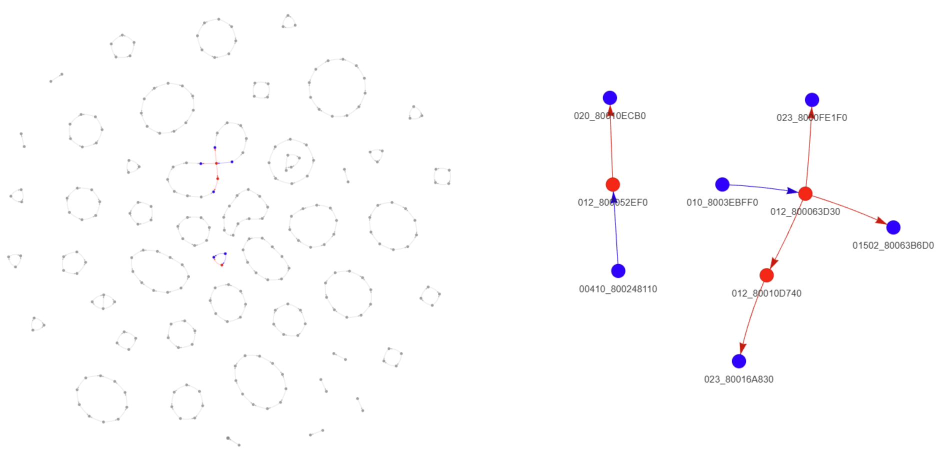


Fig. 20. Cycle pattern for only bank 012.

5 Experiment

This section describes the experiments conducted to evaluate the models on different datasets. Initially describing the experiment setup with the choice of performance metrics, and then the experimental procedure where the training and validation process are described.

5.1 Experiment Setup

5.1.1 Performance Metrics

Performance Metrics. In this research, model performance was evaluated using the F1 score, specifically focussing on the minority class. As shown in Equation 14, the F1 score is the harmonic mean of precision and recall. Given the imbalanced nature of the dataset, accuracy is not suitable as it can be misleading. In this case, with a minority clas of 0.1%, a model that classifies everything as not fraud will get a 99.9% accuracy while being completely useless. Precision (Equation 15), which measures the proportion of true positive (TP) predictions among all positive predictions (True Positives + False Positives), and recall (Equation 16), which measures the proportion of true positive (TP) predictions among all actual positives(True Positives + False Negatives), were added to provide a comprehensive evaluation of the model's performance.

$$F1 = 2 * \frac{precision * recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{14}$$

$$Precision = \frac{TP}{TP + FP} \tag{15}$$

$$F1 = \frac{TP}{TP + FN} \tag{16}$$

WandB (Weights and Biases) [48] was used to manage and track the experiments. WandB provides a platform for logging various metrics, visualising results, and facilitating hyperparameter tuning. In this research, both online and offline modes of WandB were explored for different cases. The online mode allows real-time logging and monitoring of experiments, which is useful for immediate feedback and adjustments. However, due to issues with unstable internet connections during long-running experiments, the offline mode was also used. In this mode, metrics were logged locally and then synced with the WandB server once the internet connection was stable. The use of WandB significantly improved the efficiency of the experiments, providing detailed insight into model performance and facilitating reproducibility.

5.1.2 Dealing with the unbalanced dataset. Handling the unbalanced dataset was a critical aspect of this study, given the low rate of illicit transactions compared to legitimate ones. The dataset was first loaded and preprocessed to ensure that all necessary features were included. To address the class imbalance, the CrossEntropyLoss function [34] with weighted classes was used. This approach assigns a higher penalty to misclassifications of the minority class, thus encouraging the model to pay more attention to illicit transactions, see Equation 17. The weights for the loss function were calculated based on the ratio of illicit to legitimate transactions in the dataset, see Equation 18. The goal of this method is to avoid bias towards the majority class and improving the model's ability to detect illicit transactions. Additionally, the above-mentioned use of the F1-score as the primary evaluation metric should ensure that the model's performance was accurately assessed by focussing on the (True) Positives.

The weighted cross-entropy loss function is defined by the PyTorch documentation for torch.nn.CrossEntropyLoss as:

$$\text{Loss}(x, \text{class}, w) = -\frac{1}{N} \sum_{i=1}^{N} w_{\text{class}_i} \log \left(\frac{\exp(x_i[\text{class}_i])}{\sum_j \exp(x_i[j])} \right) \tag{17}$$

where:

- N is the number of samples,
- x_i is the input for the i-th sample,
- class_i is the true class for the i-th sample,
- w_{class_i} is the weight for the class of the i-th sample.

In this research, the weights w_{class_i} were determined as:

$$w_{\text{class}_i} = \begin{cases} \frac{1}{f_{\text{positive}}} & \text{if class}_i = 1 \\ \frac{1}{f_{\text{negative}}} & \text{if class}_i = 0 \end{cases} \tag{18}$$

where f_{positive} (money laundering) and f_{negative} (no money laundering) are the frequencies of the positive and negative classes, respectively.

5.1.3 Computer resources The computational resources for this research included a PC with a 13th Gen Intel(R) Core(TM) i9-13900F CPU and an NVIDIA GeForce RTX 4060 GPU. The i9-13900F CPU handled large datasets and complex computations, and the RTX 4060 GPU significantly accelerated deep learning tasks, reducing training times for the graph neural networks. This setup ensured efficient data processing and model training.

5.2 Experimental Procedure

5.2.1 Model Training The training of the models involved running the selected baseline models, followed by testing and running various adaptations in combination with these baselines. The primary models used included GIN and GAT and MLP for classification, each configured with specific hyperparameters. The relevant hyperparameters can be found in Table 9. Most are very similar to ensure consistency between models for comparative analysis. The w_{CE2} is notably higher for the MLP model; this could be needed to handle the class imbalance, ensuring focus on the minority class.

Table 9. Important Hyperparameters of the models.

Parameter	**GIN**	**GAT**	**MLP**
Learning Rate	0.006213	0.006	0.006213
Hidden Size	66	64	66
MLP Layers	1	1	1
GNN Layers	2	2	2
Attention Heads	-	4	-
Weight CE1	1.000018	1	1.000018
Weight CE2	6.275014	6	9.23
Dropout	0.009834	0.009	0.009834
Final Dropout	0.105277	0.1	0.105277

To facilitate experimentation with different model adaptations, an argument parser was used. This parser allowed the inclusion or exclusion of specific adaptations such as those described by Egressy et al. [19], and the added time deltas, temporal encoding, and incomplete embeddings.

This configuration allowed for flexible unit testing and evaluation of each adaptation's impact on the models.

The models were trained in batches of 8192 over 100 epochs, with neighbours sampled in descending order across multiple hops. For the single-bank datasets this was increased to 1000 epochs. For neighbour sampling, LinkNeighborLoader [17] from the Pytorch Geometric library was used. This allowed for mini-batch training and constructs subgraphs based on a sampled set of input edges and the neighbouring nodes based on a parameter value of number of neighbours.

A fixed random seed was used to maintain reproducibility and tqdm logging was enabled for interactive terminal sessions.

5.2.2 Validation The validation process is critical to ensure the robustness and reliability of the models. *Temporal Train-Validation-Test Split.* As explained above, the dataset was divided into training, validation and test sets based on temporal sequences in a 60-20-20 split. This approach ensures that the model is evaluated on future data points, which simulates real-world scenarios where historical data is used to predict future events. Formally, let D be the entire dataset, t the timestamp, then:

$$D_{\text{train}} = \{x \in D \mid t(x) < t_1\} \tag{19}$$

$$D_{\text{val}} = \{x \in D \mid t_1 \leq t(x) < t_2\} \tag{20}$$

$$D_{\text{test}} = \{x \in D \mid t(x) \geq t_2\} \tag{21}$$

where t_1 and t_2 are the temporal split points.

6 Results

This section presents the results of the experiments conducted as explained in the previous section. The evaluation is done for both the full dataset and single-bank view datasets to provide comprehensive insights into the models' performance.

6.1 Full dataset

Table 10, summarises the models that were run on the full dataset and thus will be included in this research. The run times are depicted in minutes. The models include the baseline models as well as those enhanced with additional features such as Edge Updates (EU), Time Deltas (TD), Time Encoding (TE), and Incomplete Information (II).

The values in Table 10 indicate that adding the adaptations to the models greatly influences the runtime. Especially the difference between the GAT/ GIN only models and the Multi-GAT/ Multi-GIN, and the addition of edge updates (EU). Some models were excluded due to the long runtime.

Table 10. Runtime in minutes for models run on the full dataset.

	Base	**+ EU**	**+ Time Deltas**	**+ Time Encoding**	**+ Incomplete Info**	**+ EU + TD**
GAT	299	-	152	304	142	-
Multi-GAT	1616	7689	1542	-	-	-
GIN	252	-	188	147	130	-
Multi-GIN	1122	7216	1116	-	-	8229

The results of the **Graph Attention Network** show that during training, see Figure 21, the GAT+II9 (incomplete information, 9 dimensions) showed the highest performance, consistently improving throughout the epochs, while other adaptations, such as GAT+TD and GAT+TE, lagged behind. In the validation phase, see Figure 22, the baseline GAT consistently outperformed the other models, although overall F1-Scores remained low, indicating potential overfitting in training that did not generalise well to validation. In testing, see Figure 23, the results slightly improve over the epochs, but the variability is high, probably due to the overfitting. This shows the need for further tuning and perhaps more methods to improve generalisation.

The F1-Score results of the **Graph Isomorpism Network** show that during training, see Figure 24, GIN+TD quickly achieves high F1-Scores, reaching a level above 0.5, which signifies effective learning. The baseline GIN, GIN+II9, and GIN+TE models follow closely with stable scores around 0.4. In the validation phase, see Figure 25 the GIN+TD model again exhibits the highest F1-Score, reaching approximately 0.45. The standard GIN model also shows relatively stable performance, peaking around 0.4. In contrast, the GIN+II9 and GIN+TE models have lower F1-Scores, rarely exceeding 0.1. These fluctuations suggest overfitting problems with these adaptations. For testing, see Figure 26, GIN+TD continues to lead with F1-Scores consistently around 0.4-0.5. The standard GIN model performs moderately, achieving scores around 0.2-0.3. The GIN+II9 and GIN+TE models again perform poorly, with F1-Scores staying below 0.15, indicating that these adaptations might not generalize well to unseen data.

6.1.1 Best F1-Score per model. The best F1-Scores for each model on the testing section of the full dataset reveal considerable differences in performance based on the adaptions used. The Multi-GIN model combined with EU and Time Deltas achieved the highest F1-Score of 0.6871, indicating it is the most effective in this setup. Multi-GIN models generally performed well, with the base Multi-GIN achieving an F1-Score of 0.6423, and with EU alone, scoring 0.6238. For the GAT models, the highest F1-Score was 0.1959 for the baseline model. Adding any adaptations, especially time encoding, reduced its effectiveness significantly, as seen with the F1-Score of 0.04642. The Multi-GAT model also showed strong performance, particularly with time deltas, achieving an F1 score of 0.5309. The base Multi-GAT model had a solid performance with an F1-Score of 0.4614. The GIN model with time deltas performed notably well among the single models, with an F1-Score of 0.5293, suggesting that this adaptation is particularly beneficial for this model. The base GIN model achieved an F1-Score of 0.3456, while adding Incomplete Info resulted in a score of 0.1565. Overall, these results highlight the importance of model configuration, with Multi-GIN and Multi-GAT models generally outperforming their simpler counterparts, and the time encoding and incomplete information adaptations leading to overfitting.

Table 11. Best Testing F1-Score of the models run on the full dataset.

	Base	**+EU**	**+Time Deltas**	**+Time Encoding**	**+Incomplete Info**	**+EU +TD**
GAT	0.1959	-	0.1814	0.04642	0.1416	-
Multi-GAT	0.4614	0.4154	0.5309	-	-	-
GIN	0.3456	-	0.5293	0.1089	0.1565	-
Multi-GIN	0.6423	0.6238	0.611	-	-	0.6871

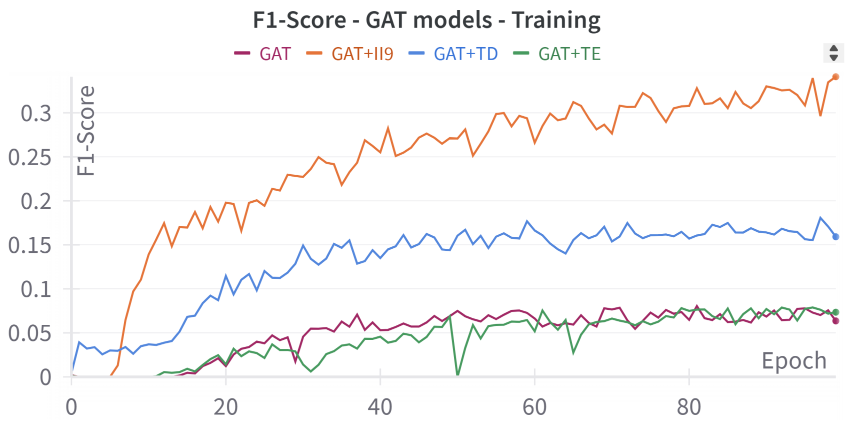


Fig. 21. Line plot of the F1-Score of the GAT models training.

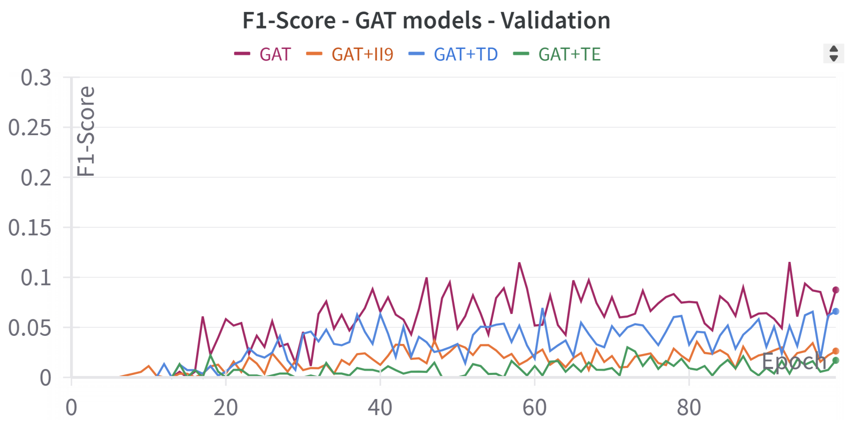


Fig. 22. Line plot of the F1-Score of the GAT models validation.

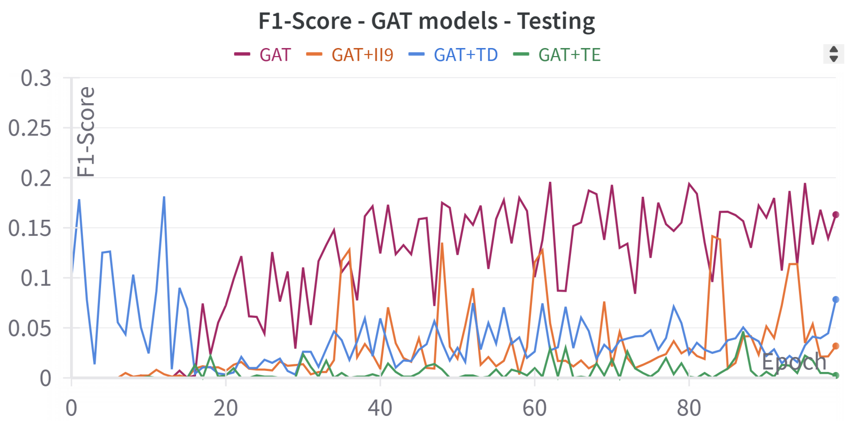


Fig. 23. Line plot of the F1-Score of the GAT models testing.

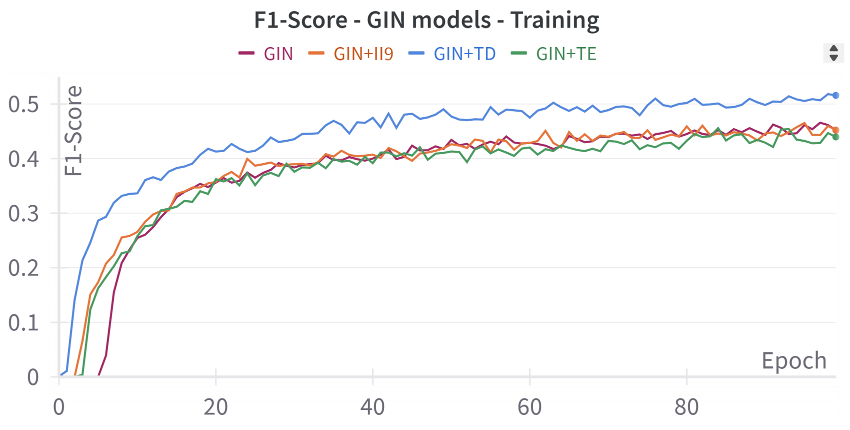


Fig. 24. Line plot of the F1-Score of the GIN models training.

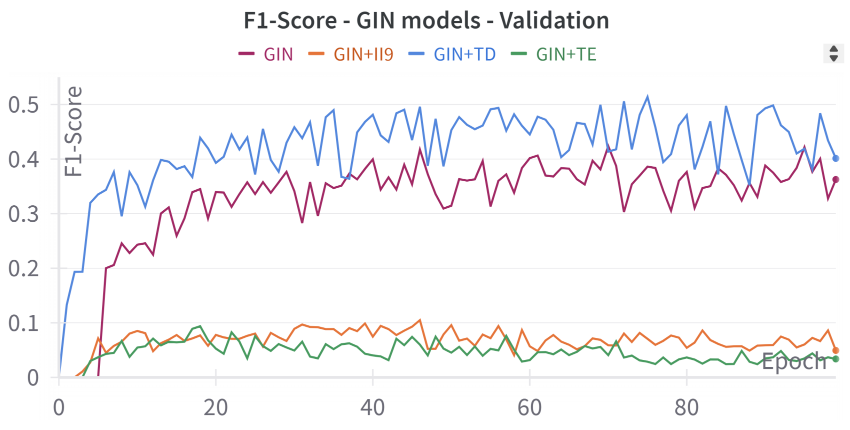


Fig. 25. Line plot of the F1-Score of the GIN models validation.

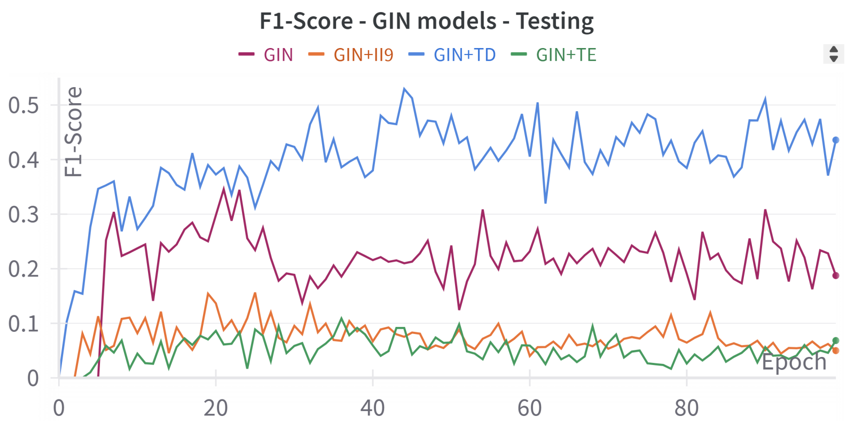


Fig. 26. Line plot of the F1-Score of the GIN models testing.

6.2 Single bank results

The single-bank view was evaluated to understand the impact of limited data visibility on model performance. Table 12 summarizes the runtime in seconds for the models run on datasets from individual banks (012 and 001). Here, the run times are much more similar, suggesting that the computation load difference is less influenced when the dataset is smaller. This could be due to the extra communication load from more batches for the full dataset.

Table 12. Runtime in seconds for models on the single-bank datasets.

	Baseline	**Time Deltas**	**Time Encoding**	**Incomplete Info**	**TD+II**	**TE+II**
GAT Bank 012	338	339	337	342	338	343
GIN Bank 012	307	317	321	371	315	369
GAT Bank 001	326	338	340	338	330	345
GIN Bank 001	311	308	318	316	307	311

The **Graph Isomorphism Network** models show varied performance in the training, validation, and testing phases. During training (Figure 27), the GIN+TE+II9 model achieves high F1-Scores quickly, stabilising around 0.95. Other adaptations, such as GIN+TD+II9 and GIN+II9 also show improvements, reaching similar F1-Scores. The baseline GIN model starts slower but eventually catches up to around 0.9. In validation (Figure 28), models with II reach double the F1 scores than those without, but overall performance is poor. During testing, see Figure 29, the GIN+TD+II9 model maintains a slight lead with F1-Scores up to 0.2, although all models perform poorly, indicating overfitting.

The **Graph Attention Network** models also show varied results. During training (Figure 30), models with the incomplete information adaption perform the best, reaching F1-Scores over 0.9. GAT+TD and GAT+TE models improve more slowly, peaking around 0.5, while the baseline GAT model struggles, only achieving around 0.3. In validation (Figure 31), the GAT+TE+II9 model slightly outperforms others with F1-Scores up to 0.35, while other models score between 0.1 and 0.2, also indicating overfitting on the limited dataset. Lastly in testing (Figure 32), the GAT+TD+II9 model leads with F1-Scores around 0.2, followed by GAT+TD and GAT+II9 models. The GAT+TE and GAT+TE+II9 models perform the worst, with scores around 0.05.

6.2.1 Best F1-Score per model. The best F1-Scores for each model on the testing section of the single-bank datasets show considerable performance variations based on the added adaptations. For GAT models on Bank 012, the highest F1-Score of 0.3226 was achieved with the combination of time deltas and incomplete information (TD + II), outperforming the baseline score of 0.1. In contrast, GIN models for the same bank reached their highest F1-Score of 0.2143 with time encoding and incomplete information (TE + II). For Bank 001, the GAT model with incomplete information alone achieved a high F1-Score of 0.5316, compared to the baseline of only 0.1633. GIN models for Bank 001 saw their maximum F1-Score improve from 0.3492 (baseline) to 0.4719 with the addition of incomplete information. These results highlight the varying impact of different adaptations, with configurations like TD+II proving particularly beneficial across different banks.

Table 13. Best Testing F1-Score of the models run with single-bank datasets.

	Baseline	**Time Deltas**	**Time Encoding**	**Incomplete Info**	**TD+II**	**TE+II**
GAT Bank 012	0.1	0.2752	0.08163	0.25	0.3226	0.2456
GIN Bank 012	0.09756	0.14	0.2029	0.1282	0.2121	0.2143
GAT Bank 001	0.1633	0.1404	0.04651	0.5316	0.5	0.386
GIN Bank 001	0.3492	0.1695	0.1918	0.4719	0.2933	0.4138

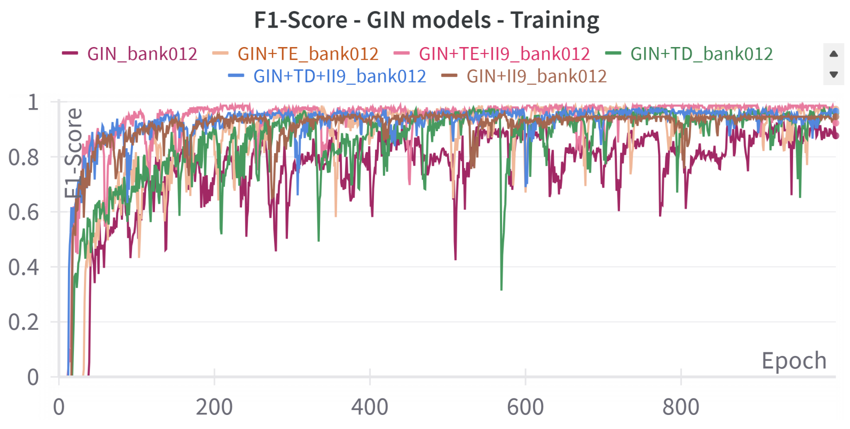


Fig. 27. Line plot of the F1-Score of the GIN models training on the single-bank view dataset.

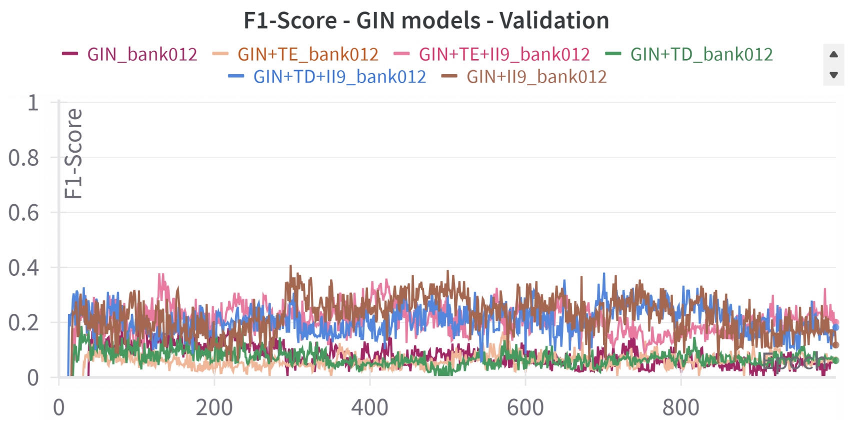


Fig. 28. Line plot of the F1-Score of the GIN models validation on the single-bank view dataset.

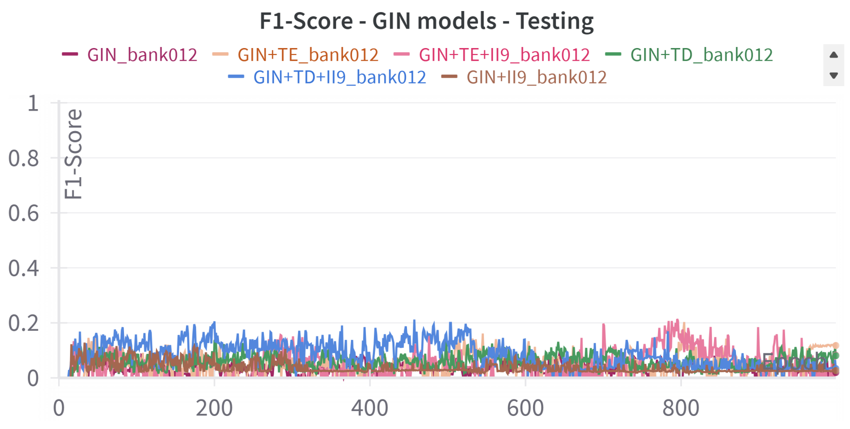


Fig. 29. Line plot of the F1-Score of the GIN models testing on the single-bank view dataset.

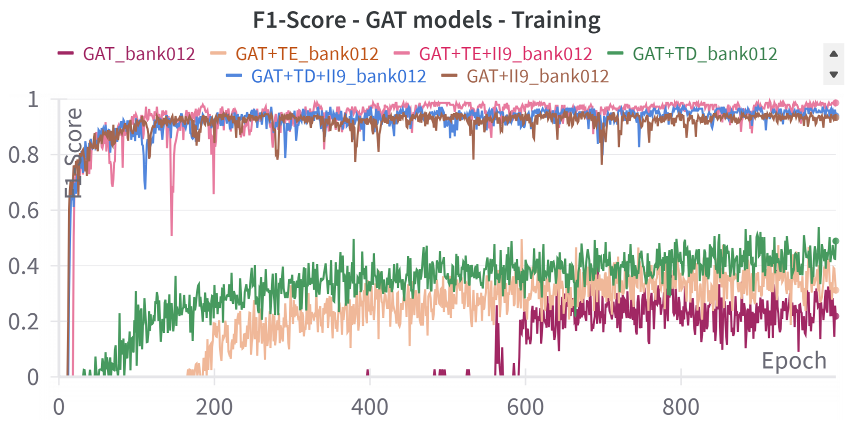


Fig. 30. Line plot of the F1-Score of the GAT models training on the single-bank view dataset.

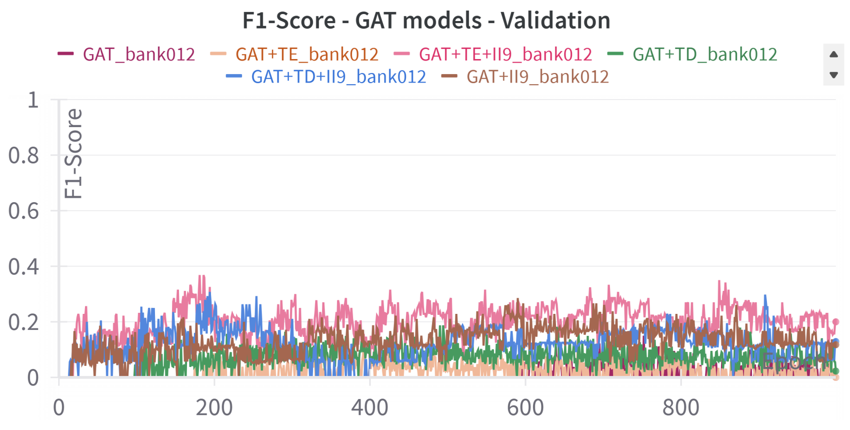


Fig. 31. Line plot of the F1-Score of the GAT models validation on the single-bank view dataset.

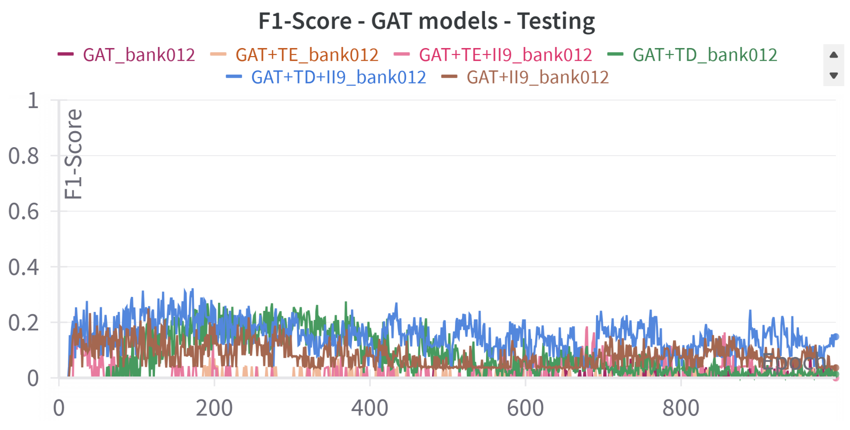


Fig. 32. Line plot of the F1-Score of the GAT models testing on the single-bank view dataset.

7 Discussion

Based on the analysis on the dataset that represents the single-bank view and the experiments with the graph neural networks and added adaption, this section will discuss the results in the broader context of the problem.

7.1 Comparison of Full Dataset and Single Bank Results

The single-bank view analysis underscored the significant challenges posed by limited data visibility. Visualising and training the data from the perspective of individual banks (in this case Bank 012 and Bank 001) revealed substantial fragmentation of the laundering patterns. Here, two main conclusions can be drawn:

Pattern Disruption: Complex laundering patterns such as gather-scatter and fan-in often appear incomplete or entirely lost when viewed from a single bank's perspective. This fragmentation complicates the detection of suspicious activities and increases the risk of both false positives and false negatives.
Model Performance: The models trained on single-bank datasets exhibited overfitting issues, with high training F1 scores but significantly lower validation and testing scores.

The comparison with the full dataset was made to support the conclusions made from the single-bank view. The overfitting could, and partially does, indicate that the time encoder adaption overfits the model and does not improve performance. However, the incomplete information adaption shows poor performance on the full dataset but improves the F1-score when working with the partial networks.

This is mainly for comparative purposes, since the incomplete data overfits and performs poorly overall. The conclusion can be drawn that, in addition to overfitting, the models are mainly trained incorrectly. The labelled patterns no longer represent the patterns that they are labelled as. Therefore, the model might learn the correct representation that is fed, but not the correct representation of the fraudulent patterns.

7.2 Implications for AML Detection

The findings highlight the significant challenges for AML detection when not able to share transaction data amongst banks. Machine learning models require the majority of the data to accurately identify suspicious patterns. Incomplete transaction data, as seen in the single-bank view, leads to partial and reduced laundering patterns that models struggle to interpret. This leads to the conclusion that some form of sophisticated data-sharing mechanisms or legislative changes are required to allow broader data access in order to enhance the effectiveness of AML efforts.

8 Conclusion and Further Research

This research highlights the critical importance of having access to comprehensive transaction data for effective anti-money laundering (AML) detection using machine learning models. Models trained on the complete dataset, in this case Multi-GNN models such as Multi-GIN with edge updates and time deltas, significantly outperformed those with limited single-bank data. The findings emphasise the need for improved data sharing frameworks and legislative support to enhance AML detection capabilities. Addressing the limitations and opportunities related to data visibility, temporal modelling, and incomplete information adaptions is essential to improve transaction monitoring systems and reduce labour costs.

The **challenges and limitations** during this research revolved around access to a representative dataset, finding transparant previous research and implementations, and the high computational demands of training complex GNN models on large datasets.

- Quite early on it was clear that real transaction data could not be shared and used for this research. The search for a suitable synthetic dataset was complicated by the requirements for this research. The dataset needed to contain recognisable patterns, multiple banks, and a large volume of labelled transactions. With the synthetic nature of the dataset comes that, while useful for research, it may not fully capture the complexities of real-world financial transactions. The experiment's reliance on predefined laundering patterns also limits its ability to generalise findings to more diverse or evolving money laundering tactics.
- To be able to focus on the addition of components to the models and perform comparative research, an existing baseline was required. These models needed to be properly supported by papers and have an implementation available online (e.g., on GitHub). The challenge was the limited number of published implementations and the quality of those available. Many implementations contained errors, and the results described in the papers were difficult to reproduce accurately.
- Training on a dataset of over 5 million records with heavy-to-train graph neural network models could take up to a week. Some models (PNA) took more than four hours per epoch, resulting in an estimated computational time of almost three weeks.

The findings of this research highlight several critical areas for **further research and potential improvements**:

- Integration of data across financial institutions: Waiting for legislative changes to allow more comprehensive data sharing between banks could enhance the ability to detect money laundering tactics that are dispersed. However, a more active approach would be to study the possibilities of sharing data more privately. An opportunity could lie in the incomplete information adaption, where the broader structural information is added as embeddings. If financial institutions could share just the graphical data of anonimized nodes and edges, another institution (e.g., TMNL) would combine there graphs with the other banks (they would have no data even remotely referring to account holders). Then they would make the structural representation (incomplete information adaption) embeddings of the full picture and share this with the individual financial institutions.
- Enhanced Temporal Modelling: Further research into temporal adaptations that can better capture the timing and sequence of transactions is necessary. This includes exploring more sophisticated time encoding techniques that can better generalise.
- Explainability and Transparency: Ensuring that ML models used in AML efforts are explainable is critical for regulatory compliance and stakeholder trust. Future work could focus on integrating advanced explainability techniques to make model decisions more interpretable.

The significant drop in performance with single-bank data highlights the limitations imposed by data privacy regulations. Sophisticated ML algorithms alone are not sufficient to mitigate these limitations; some form of data sharing is necessary.

If a real-world dataset were available, the next steps would involve validating the current findings against real-world data to assess the generalizability and practical applicability of the models. This would include focusing on the incomplete information adaptation, which has shown promise in the context of the single-bank view. By validating these models with actual transaction data, researchers could fine-tune the models to address any emerging challenges and further enhance their effectiveness.

The focus would then shift to exploring anonymised data-sharing methods, such as embedding-based approaches, which could provide a way to enhance AML detection while adhering to legal and privacy regulations. Such methods would allow financial institutions to share structural transaction data without compromising individual privacy, potentially leading to more robust and comprehensive AML systems.

In conclusion, while synthetic datasets provide a useful starting point, the availability of real-world data is crucial for advancing AML detection capabilities. By addressing the limitations highlighted in this research and focusing on practical implementation, the AML community can move towards more effective and efficient transaction monitoring systems.

References

1. Erik Altman, Beni Egressy, Jovan Blanuvsa, and Kubilay Atasu. Realistic synthetic financial transactions for anti-money laundering models. *ArXiv*, abs/2306.16424, 2023. URL: `https://arxiv.org/abs/2306.16424`.
2. Federico Baldassarre and Hossein Azizpour. Explainability techniques for graph convolutional networks. *CoRR*, abs/1905.13686, 2019. URL: `http://arxiv.org/abs/1905.13686`, `arXiv:1905.13686`.
3. Banken.nl. Abn amro schikt voor 480 miljoen om witwasfraude. URL: `https://www.banken.nl/nieuws/23019/abn-amro-schikt-voor-480-miljoen-om-witwasfraude`.
4. Banken.nl. Brussel zet streep door huidige opzet transactiemonitoring nederland. URL: `https://www.banken.nl/nieuws/25106/brussel-zet-streep-door-huidige-opzet-transactiemonitoring-nederland`.
5. Banken.nl. Dnb tikt rabobank opnieuw op vingers vanwege tekortkomende witwascontroles. URL: `https://www.banken.nl/nieuws/23416/dnb-tikt-rabobank-opnieuw-op-vingers-vanwege-tekortkomende-witwascontroles`.
6. Banken.nl. Dnb volksbank ernstig tekortgeschoten in wwft verplichtingen. URL: `https://www.banken.nl/nieuws/24725/dnb-volksbank-ernstig-tekortgeschoten-in-wwft-verplichtingen`.
7. Banken.nl. Ing schikt voor €775 miljoen om faciliteren witwassen. URL: `https://www.banken.nl/nieuws/21124/ing-schikt-voor-775-miljoen-om-faciliteren-witwassen`.
8. Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, 2018. URL: `https://arxiv.org/abs/1806.01261`, `arXiv:1806.01261`.
9. Mário Cardoso, Feedzai Pedro Saleiro, and Feedzai Pedro Bizarro. Laundrograph: Self-supervised graph representation learning for anti-money laundering. *Proceedings of the Third ACM International Conference on AI in Finance*, 2022. `doi:10.1145/3533271.3561727`.
10. Zhiyuan Chen, · Le Dinh, Van Khoa, · Ee, Na Teoh, · Amril Nazir, · Ettikan, Kandasamy Karuppiah, · Kim, and Sim Lam. Machine learning techniques for anti-money laundering (aml) solutions in suspicious transaction detection: a review. *Knowledge and Information Systems*, 57:245–285, 2018. `doi:10.1007/s10115-017-1144-z`.
11. Cmach. GitHub - CMACH508/DynGAT, 2024. URL: `https://github.com/CMACH508/DynGAT`.
12. McKinsey & Company. Transforming approaches to aml and financial crime, 2019. URL: `https://www.mckinsey.com/~/media/McKinsey/Business%20Functions/Risk/Our%20Insights/Transforming%20approaches%20to%20AML%20and%20financial%20crime/Transforming-approaches-to-AML-and-financial%20crime-vF.pdf`.
13. Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *CoRR*, 2020. URL: `https://arxiv.org/abs/2004.05718`, `arXiv:2004.05718`.
14. José de Jesús Rocha-Salazar, María Jesús Segovia-Vargas, and María del Mar Camacho-Miñano. Money laundering and terrorism financing detection using neural networks and an abnormality indicator. *Expert Systems with Applications*, 169, 5 2021. `doi:10.1016/j.eswa.2020.114470`.
15. Xinwei Deng, V. Roshan Joseph, Agus Sudjianto, and C. F. Jeff Wu. Active learning through sequential design, with applications to detection of money laundering. *Journal of the American Statistical Association*, 104(487):969–981, 2009. URL: `http://www.jstor.org/stable/40592268`.
16. Read The Docs. Pyg documentation — pytorch geometric documentation. URL: `https://pytorch-geometric.readthedocs.io/en/latest/index.html`.
17. Read The Docs. Torch_geometric.loader.link_neighbor_loader — PyTorch_geometric documentation. URL: `https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/loader/link_neighbor_loader.html`.
18. Read The Docs. Pyvis 0.1.3.1 documentation, 2018. URL: `https://pyvis.readthedocs.io/en/latest/tutorial.html`.
19. Béni Egressy, Luc von Niederhäusern, Jovan Blanusa, Erik Altman, Roger Wattenhofer, and Kubilay Atasu. Provably powerful graph neural networks for directed multigraphs. *AAAI Conference on Artificial Intelligence*, 2024. `arXiv:2306.11586`.
20. William L. Hamilton. *Graph Representation Learning.* Springer International Publishing, 2020. `doi:10.1007/978-3-031-01588-5`.

21. Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning.* Springer New York, 2009. doi:10.1007/978-0-387-84858-7.
22. IBM. GitHub - IBM/Multi-GNN: Multi-GNN Architectures for Anti-Money laundering. URL: https://github.com/IBM/Multi-GNN.
23. Jiachui Jing. Research on node classification algorithm based on deep network representation learning under incomplete information. *Proceedings - 2023 2nd International Conference on 3D Immersion, Interaction and Multi-Sensory Experiences, ICDIIME 2023*, pages 131–135, 2023. doi:10.1109/ICDIIME59043.2023.00031.
24. Martin Jullum, Anders Løland, Ragnar Bang Huseby, Geir Ånonsen, and Johannes Lorentzen. Detecting money laundering transactions with machine learning. *Journal of Money Laundering Control*, 23(1):173–186, January 2020. URL: https://ideas.repec.org/a/eme/jmlcpp/jmlc-07-2019-0055.html, doi:10.1108/JMLC-07-2019-0055.
25. Kaggle. Discussion - stack and bipartite patterns - it-aml dataset. URL: https://www.kaggle.com/datasets/ealtman2019/ibm-transactions-for-anti-money-laundering-aml/discussion/458775.
26. Dattatray Vishnu Kute, Biswajeet Pradhan, Nagesh Shukla, and Abdullah Alamri. Deep learning and explainable artificial intelligence techniques applied for detecting money laundering-a critical review, 2021. doi:10.1109/ACCESS.2021.3086230.
27. Yuebing Liang, Zhan Zhao, and Lijun Sun. Dynamic spatiotemporal graph convolutional neural networks for traffic data imputation with complex missing patterns. *CoRR*, abs/2109.08357, 2021. URL: https://arxiv.org/abs/2109.08357, arXiv:2109.08357.
28. Yuebing Liang, Zhan Zhao, and Lijun Sun. Memory-augmented dynamic graph convolution networks for traffic data imputation with diverse missing patterns. *Transportation Research Part C: Emerging Technologies*, 143:103826, 10 2022. doi:10.1016/J.TRC.2022.103826.
29. Chen Liu, Ziran Li, and Lixin Zhou. Missing nodes detection for complex networks based on graph convolutional networks. *Journal of Ambient Intelligence and Humanized Computing*, 14:9145–9158, 7 2023. doi:10.1007/s12652-022-04418-3.
30. FIU Nederland. Meldergroepen. URL: https://www.fiu-nederland.nl/home/meldergroepen/.
31. Berkan Oztas, Deniz Cetinkaya, Festus Adedoyin, Marcin Budka, Huseyin Dogan, and Gokhan Aksu. Perspectives from experts on developing transaction monitoring methods for anti-money laundering. *2023 IEEE International Conference on e-Business Engineering (ICEBE)*, pages 39–46, 2023. URL: https://api.semanticscholar.org/CorpusID:266393514, doi:10.1109/icebe59045.2023.00024.
32. Berkan Oztas, Deniz Cetinkaya, Festus Fatai Adedoyin, and Marcin Budka. Enhancing transaction monitoring controls to detect money laundering using machine learning. *2022 IEEE International Conference on e-Business Engineering (ICEBE)*, pages 26–28, 2022. URL: https://api.semanticscholar.org/CorpusID:256669414, doi:10.1109/ICEBE55470.2022.00014.
33. Phillip E. Pope, Soheil Kolouri, Mohammad Rostami, Charles E. Martin, and Heiko Hoffmann. Explainability methods for graph convolutional neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10764–10773, 2019. doi:10.1109/CVPR.2019.01103.
34. PyTorch. CrossEntropyLoss — PyTorch 2.3 documentation. URL: https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html.
35. Rabobank. Know your customer (kyc). URL: https://www.rabobank.nl/particulieren/service/kyc.
36. Rijksoverheid. Wat betekent de wwft voor bedrijven en klanten? — financiële sector. URL: https://www.rijksoverheid.nl/onderwerpen/financiele-sector/aanpak-witwassen-en-financiering-terrorisme/veelgestelde-vragen-wwft#anker-3-wat-ben-ik-verplicht-te-doen-volgens-de-wwft.
37. Ahmad Salehi, Mehdi Ghazanfari, and Mohammed Fathian. Data mining techniques for anti money laundering, 2017. URL: http://www.ripublication.com.
38. Angela Samantha Maitland Irwin, Kim-Kwang Raymond Choo, and Lin Liu. An analysis of money laundering and terrorism financing typologies. *Journal of Money Laundering Control*, 15(1):85–111, 2011. doi:10.1108/13685201211194745.
39. Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web*, pages 593–607, 2018. URL: https://arxiv.org/abs/1703.06103, arXiv:1703.06103, doi:10.1007/978-3-319-93417-4_38.
40. Friedrich Schneider and Ursula Windischbauer. Money laundering: some facts. *European Journal of Law and Economics*, 26(3):387–404, 2008. doi:10.1007/s10657-008-9070-x.

41. Stefan Studer, Thanh Binh Bui, Christian Drescher, Alexander Hanuschkin, Ludwig Winkler, Steven Peters, and Klaus-Robert Müller. Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology. *Machine learning and knowledge extraction*, 3(2):392–413, 4 2021. `doi:10.3390/make3020020`.
42. Suzumura and Kanezashi. It-aml - anti-money laundering datasets. URL: `https://github.com/IBM/AMLSim`.
43. Hibiki Taguchi, Xin Liu, and Tsuyoshi Murata. Graph convolutional networks for graphs containing missing features. *Future Generation Computer Systems*, 117:155–168, 4 2021. `doi:10.1016/J.FUTURE.2020.11.016`.
44. DGL Team. Welcome to Deep Graph Library Tutorials and Documentation — DGL 2.3 Documentation, 2018. URL: `https://docs.dgl.ai/en/latest/index.html`.
45. Pavlo Tertychnyi, Tommy Lindström, Changling Liu, and Marlon Dumas. Detecting group behavior for anti-money laundering with incomplete network information. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 2383–2388, 2022. `doi:10.1109/BigData55660.2022.10020321`.
46. TMNL. Transaction Monitoring Netherlands adapts its working method to new European legislation - TMNL, 7 2024. URL: `https://tmnl.nl/en/article/transaction-monitoring-netherlands-adapts-its-working-method-to-new-european-legislation/`.
47. Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio', and Yoshua Bengio. Graph attention networks. *ArXiv*, abs/1710.10903, 2017. URL: `https://api.semanticscholar.org/CorpusID:3292002`.
48. WandB. W&B Docs — Weights & Biases Documentation. URL: `https://docs.wandb.ai/`.
49. Tianpeng Wei, Biyang Zeng, Wenqi Guo, Zhenyu Guo, Shikui Tu, and Lei Xu. A dynamic graph convolutional network for anti-money laundering. In De-Shuang Huang, Prashan Premaratne, Baohua Jin, Boyang Qu, Kang-Hyun Jo, and Abir Hussain, editors, *Advanced Intelligent Computing Technology and Applications*, pages 493–502, Singapore, 2023. Springer Nature Singapore.
50. Wetten.nl. Regeling - wet ter voorkoming van witwassen en financieren van terrorisme - bwbr0024282. URL: `https://wetten.overheid.nl/BWBR0024282/2022-11-01`.
51. Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *CoRR*, 2019. URL: `https://arxiv.org/abs/1810.00826`, `arXiv:1810.00826`.
52. Zi Yi, Xinwei Cao, Zuyan Chen, and Shuai Li. Artificial intelligence in accounting and finance: Challenges and opportunities. *IEEE Access*, 11:129100–129123, 2023. URL: `https://api.semanticscholar.org/CorpusID:265266862`, `doi:10.1109/ACCESS.2023.3333389`.
53. Jingwei Zuo, Karine Zeitouni, Yehia Taher, and Sandra Garcia-Rodriguez. Graph convolutional networks for traffic forecasting with missing values. *Data Mining and Knowledge Discovery*, 37(2):913–947, 2022. `doi:10.1007/s10618-022-00903-7`.

Appendix A Graphs patterns for only bank 012.

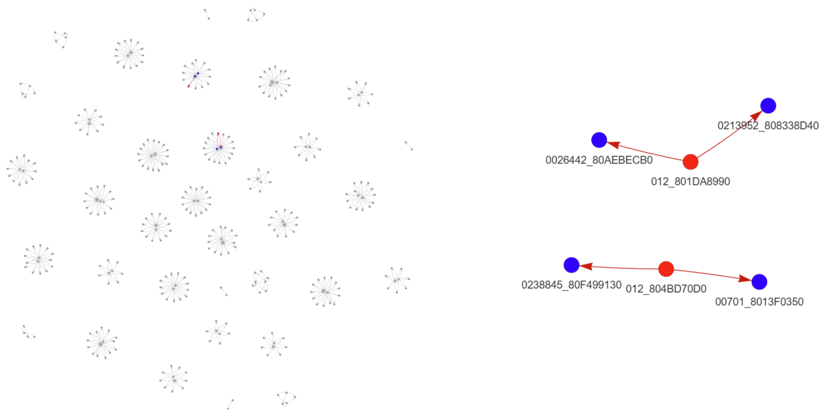


Fig. 33. Scatter-Gather pattern for only bank 012.

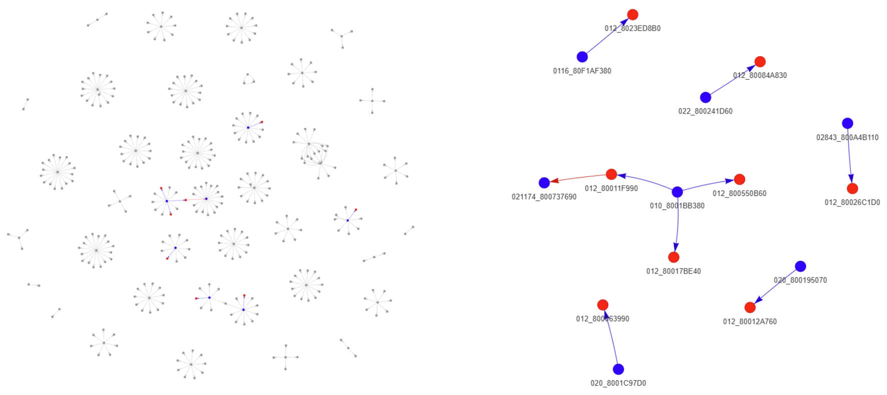


Fig. 34. Fan-out pattern for only bank 012.

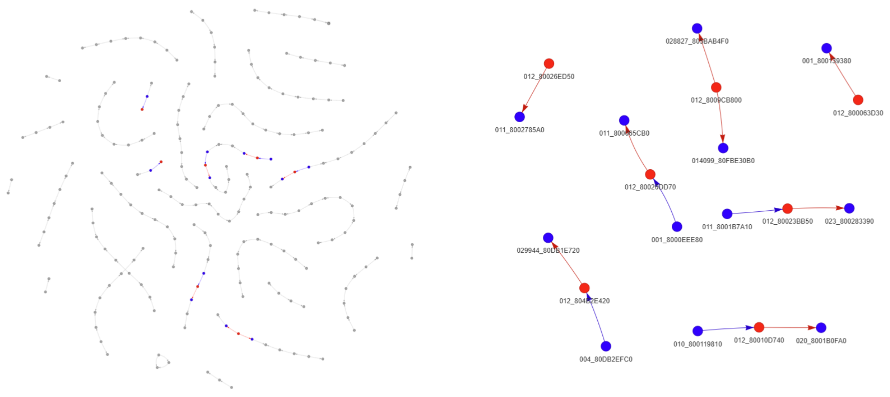


Fig. 35. Random pattern for only bank 012.