POLITECNICO DI TORINO

Master Degree course in Computer Science:
Artifical Intelligence and Data Analytics

Master Degree Thesis

Network Analysis Enhancements to improve Anti Money Laundering Transaction Monitoring Systems in Banks

Supervisors

Prof. Marco Mellia
Dario Moncalvo , PhD

Candidate
Stefano Barcio

Academic Year 2024/2025

Summary

Financial Crime poses a persistent and evolving threat to the integrity of the global financial system, especially in the forms of money laundering and terrorist financing. To combat these offences, financial institutions, government regulatory bodies, law enforcment and international agencies coordinate their efforts, sharing expertise and innovative technologies. The ensemble of said knowledge and technologies goes under the name of Anti Financial Crime (AFC). However, as banking institutions handle ever-growing volumes of transactions, their traditional, rule-based AFC monitoring pipelines strain under the scale and complexity of modern laundering schemes. Banks like Intesa Sanpaolo need to manage millions of monthly wire transfers, thus exposing the limitations of current transaction-monitoring systems. Moreover, particularly problematic are, in general, the high false-positive rates in suspiscious activity reports, that burden the downstream monitoring pipeline and divert experts' attention from truly suspicious activity. This misallocation not only slows investigations, but also increases the risk that sophisticated illicit networks remain undetected.

To address these challenges, this work introduces a dual, AI-driven approach based on Network Analysis (NA) principles to improve the efficiency of the Transaction Monitoring (TXM) section of the Intesa Sanpaolo's AFC pipeline. Representing transactions as directed graphs, where each account or institution is a node and each transfer an edge, allows to leverage powerful principles of graph theory, community detection, and graph temporal deep learning. This work proposes two innovative models, **TRACED** and **GRAND-Net**, each one starting from two complementary perspectives, distinguished by their scale of application.

TRACED (Temporal Relational Analysis for Criminal Entities Detection) employs a bottom-up strategy, starting from the transaction history of one—or a small group—of ISP clients over a given period of time. These transactions, initially stored in raw Excel files, are ingested and mapped into a directed graph using standard libraries. This allows to extract topological and temporal features of each client that were hidden and difficult to detect in the original Excel data. Starting from this network, advanced Network Analytics and machine-learning techniques are applied to reconstruct potential criminal subgraphs among the clients' counterparties, expanding the initial investigative perimeter. TRACED produces two innovative visualizations: an interactive cumulative-balance chart

that highlights anomalous transaction patterns and a counterpart-centric timeline that reveals temporal flows. TRACED is also equipped with another novel proposal of this work: a community-detection algorithm — **TRACED-FastMaN** — that isolates clusters of transactions correlated by amount, timing, or frequent co-occurrence. By feeding these clusters into a linear regression model, TRACED computes an automated ML-based risk score for each client. The combination of rich visual outputs and quantitative scoring provides analysts with a concise, combined qualitative and quantitative overview of each client's transactional risk, allowing to automate the exploration of possible criminal networks starting from a single suspect, and to capture deeper, multi-step laundering patterns that conventional methods often miss.

GRAND-Net (Graph Recurrent ANomaly Detection Network) adopts a complementary top-down approach that contrasts with TRACED's bottom-up focus. It employs a graph-recurrent neural network trained to detect anomalies in large-scale, macroscopic financial flows. Starting from the complete network spanned by the totality of transactions managed by Intesa Sanpaolo over a defined period, GRAND-Net leverages an innovative dual-head reconstruction architecture to learn robust relational, topological and temporal embeddings—capturing both inter-node interactions and their evolution over time. Using these embeddings, the model forecasts the network's structure for future intervals. This reconstructed graph can then support various downstream tasks; in this thesis, we explore its use for edge-level anomaly detection by validating on test graphs augmented with synthetic anomalies derived from documented money-laundering and terrorist-financing patterns. By isolating anomalous subgraphs, GRAND-Net assigns an anomaly score to each edge, empowering analysts to flag unusual high-volume flows or abrupt macroeconomic shifts—without relying on predefined suspects or rigid rule-based scenarios. Importantly, GRAND-Net is trained in a fully unsupervised manner, eliminating the need for transaction-level labels—a common constraint in existing financial anomaly-detection models. Additionally, leveraging existing graph downsampling algorithms applied to transactional data, and applying principles of transfer learning and fine-tuning, GRAND-Net enables training on a much smaller subset of nodes and edges, drastically reducing time and memory requirements while still supporting full-scale inference on future networks with no loss in accuracy or performance.

Both models have been tested on real data taken from ISP proprietary databases, and exhibit superior performance with respect to existing state of the art related models. Particularly, TRACED achieves an AUC of 0.86 in separating legitimate clients from accounts used to perform money laundering operations, and GRAND-Net outperforms static graph baselines (including DeepWalk and spectral clustering) by large margins, achieving edge-level AUCs above 0.93 and robust performance even as the fraction of injected anomalies rises up to twenty percent of total network's edges.

A key contribution of this thesis is models' integration in existing AFC systems. Both TRACED and GRAND-Net have been deployed within Intesa Sanpaolo's existing Transaction Monitoring pipeline without requiring fundamental changes to the IT infrastructure. TRACED leverages the bank's scheduled SAS exports, producing interactive dashboards

that slot into analysts' web portals. GRAND-Net operates in ISP's cloud-based GPU environment, retraining on rolling windows of 24–36 months and scoring fresh monthly basetables to generate edge-level alerts. Analysts teams can now consume these AI-powered insights alongside traditional rule-based alerts, prioritizing investigations by risk score or anomaly level rather than arbitrary thresholds.

The practical impact of the proposed models in the ISP's TXM pipeline will be evaluated in the months following the publication of this work. However, an estimate of its contributions can be performed by looking at the features and technological innovations implemented. First, by reducing the reliance on rigid, rule-based scenarios and flagging only those accounts or flows with high-confidence AI scores, the system goes in the direction of achieving a significant drop in false positives—freeing compliance officers to focus on genuinely suspicious cases. Second, the combination of detailed visualizations and quantitative risk/anomaly scores accelerates the investigation process: this will allow analysts to better face the observed annual increase in money-laundering and terrorism financing related offences, making the overall TXM pipeline more solid and able to manage an higher number of cases.

This thesis outlines several avenues for expansion. Incorporating external data sources, such as sanctions lists or corporate registries, could enrich node attributes and further improve detection. In the same way, training the two algorithms on longer, more complex data available in ISP's internal databases will allow to achieve a more robust embedding and better generalization, tailoring the models to the precise ISP's needs and tasks, hopefully improving accuracy results on real money laundering and terrorism financing cases.

Network Analysis tools are still far from being fully exploited in the AFC field, and combining AI and NA is a further level of progress that, in the AFC domain, presents still limited research literature and industrial experiences. This work goes in the direction of building one step more in this process, and finally brings two novel, proprietary algorithms to the field of NA in AFC. Together, they transform terabytes of raw transaction data into prioritized, investigator-ready intelligence—strengthening the bank's defenses against the increasingly sophisticated and globalized threat of financial crime.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction to Anti Financial Crime

1.1 What is Financial Crime

The expression Financial Crime refers to all kinds of illicit activities targeting financial and economic systems, tools, and operations, with the goal of obtaining unlawful benefits—either through direct economic gain or by securing indirect advantages for the perpetrators.

There are many forms of financial crime; the ones that this work will deal with are Money Laundering (ML) and Terrorism Financing. Money Laundering is the process which aims to disguise illegally obtained funds through a number of seemingly lawful operations, such as structuring deposits, routing money through multiple accounts, and investing in legitimate-looking assets. Those operations are designed to confuse and deceive authorities and make dirty money appear clean. Terrorist Financing, on the other hand, is the provision of financial resources—whether obtained through legal or illicit means—to support terrorist organizations or activities. These examples represent only a subset of financial-related offenses, whose number and complexity continue to grow with the emergence of new financial instruments and technologies that introduce novel risks and opportunities for abuse [1].

Indeed, the digitalization of financial systems has given rise to new and sophisticated threats, making financial crime harder to detect and prevent. The emergence of cryptocurrencies and blockchain technologies has enabled anonymous transactions that can be exploited for money laundering and terrorist financing. Additionally, cyber-enabled financial crimes such as ransomware attacks, phishing schemes, and deepfake fraud have become more prevalent, allowing criminals to exploit vulnerabilities in digital banking, payment systems, and online transactions

[2] [3]. The use of artificial intelligence and machine learning has also contributed to financial crime, with advanced algorithms being used to automate fraud, manipulate financial markets, or evade detection by compliance systems.

Organized crime groups, corrupt officials, and even individuals may perpetrate these offenses, leveraging loopholes in regulatory frameworks or unsufficient monitoring of financial entities to achieve their objectives. To reduce the risk posed by these threats, it is crucial for all involved entities to build an encompassing, multi-disciplinary infrastructure of legal enforcement, advanced technological tools and robust international cooperation to trace and recover illicitly obtained assets [4].

1.1.1 Focus on Money Laundering

Money Laundering (ML), among the various financial crime (FC) offenses, plays a particularly significant role in the context of this work. This is the process of legitimizing funds obtained from illegal activities, with the aim of concealing their original source and integrating them into the legitimate economic system. It is a criminal activity that typically originates from another illegal act, such as drug trafficking, fraud, or corruption. Money Laundering generally requires three distinct stages: *placement*, *layering*, and *integration* [5].

During the placement stage, illicit funds are introduced into the financial system through methods such as falsified documentation or currency exchanges, masking the true origin of the money with deceptive operations [6]. In the layering stage, the funds are further obscured by moving them through a series of seemingly legitimate transactions, which serve no purpose other than to disguise their source and hinder detection by law enforcement and intelligence agencies (LEAs). This often involves the use of shell companies and cross-border transfers [7] [8]. Finally, in the integration stage, the laundered funds are reintroduced into the economy as legitimate assets, often through investments or cash-intensive businesses such as restaurants and bars [9].

Along this line, money laundering represents a critical challenge for the global financial system, with far-reaching economic and social implications. For this reason, it is important to understand the impact of this phenomenon in terms of the total amount diverted from the legitimate economy. The complexity of measuring this impact stems from the wide-ranging nature of predicate offenses and the diversity of consequences associated with each laundered amount. While various estimates have emerged in recent years, methodological limitations and data inconsistencies have led to divergent findings. Nevertheless, the collective evidence suggests the scale of money laundering is substantial enough to warrant significant public policy intervention. Research by the United Nations Office on Drugs and Crime [10], as documented in their analysis of multiple studies, suggests that annual

losses attributable to money laundering activities range from approximately 2% to 5% of global GDP. Similar numbers are found in the italian landscape, with estimates accounting for a loss of approximately 1.8% of GDP between 2018 and 2022 [11].

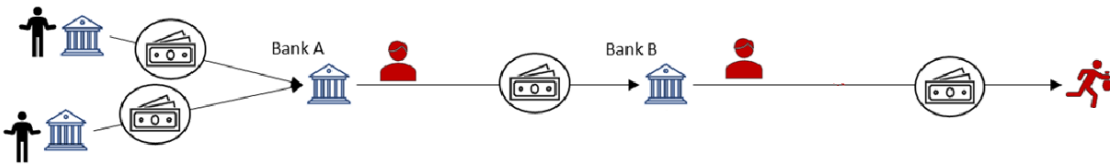


Figure 1.1: Practical illustration of the three-stage money laundering process: funds obtained through fraudulent activities are deposited in Bank A, transferred to Bank B to obscure their illicit origin, and finally withdrawn via ATM.

1.2 The Anti-Financial Crime: an Overview

As defined by the Financial Action Task Force (FATF)[1], the Anti-Financial Crime comprises measures to protect financial systems and the broader economy from the threats of money laundering and terrorist financing. It encompasses the full spectrum of anti-money laundering (AML) and counter-financing of terrorism (CFT) measures designed to safeguard the integrity of the international financial system by preventing, detecting and disrupting illicit financial flows.

Financial institutions (FIs) operate at the front line and must implement different levels of preventive controls:

- **Know Your Customer (KYC):** identifying and assessing the money-laundering and terrorist-financing risks posed by each client;
- **Transaction Monitoring (TXM):** ongoing scrutiny of transactions to detect unusual or suspicious patterns;
- **Financial Sanctions (FS):** freezing the assets of designated individuals and entities under international mandates.

When an FI detects or suspects illicit activity, it files a report with its national Financial Intelligence Unit (FIU). The FIU serves as the centralized agency for receiving, analyzing and disseminating financial intelligence [12]. Upon identifying sufficient evidence of wrongdoing, the FIU refers cases to law enforcement agencies (LEAs) or public prosecutors for investigation and prosecution.

[1] `https://www.fatf-gafi.org/en/home.html`

As previously discussed and also stated in the annual italian FIU report [4], money launderers greatly benefit from the discrepancies and inconsistencies between regulatory frameworks across different jurisdictions to further their illicit activities. These differences create significant opportunities for criminals to shift operations to countries with weaker regulatory oversight or more permissive legal environments. By doing so, they can more easily evade detection and avoid the more robust monitoring systems found in other jurisdictions. Additionally, the use of cross-border transactions allows money launderers to fragment their operations, making it increasingly difficult for investigators to track the flow of illicit funds. This helps criminals to bypass nation-specific law enforcement agencies (LEAs) and financial intelligence units (FIUs), hindering their efforts to build a clear, unified picture of the overall criminal operation .

Recognizing these challenges, the international community has made significant progress in recent decades to combat money laundering by enhancing cooperation and collaboration among countries [12]. Over the last century, efforts have intensified to establish a more unified global framework aimed at coordinating the actions of law enforcement agencies in the context of Anti-Money Laundering (AML) operations [13].

Building on this framework, in more recent years there has been the need to categorize activities related to combating the financial support of terrorist organizations under a distinct terminology. These activities, while not always falling strictly under the definition of money laundering—since the origin of the funds may not necessarily be illegal— however share many characteristics and behaviors with typical money laundering operations, while also requiring similar investigative and preventive approaches. As a result, the term Counter Financing of Terrorism (CFT) emerged, which is now commonly used alongside Anti-Money Laundering (AML) to describe the full scope of efforts to address both money laundering and the financial networks supporting terrorism.

1.3 Key Institutions and Organizations in the Fight Against Financial Crime

Starting from a global perspective and narrowing down to the Italian context, this paragraph will explore the different players responsible for AML/CFT provisions at different levels.

1.3.1 The Financial Action Task Force (FATF)

The Financial Action Task Force (FATF) [2] is an international standards setting body, considered as "the United Nations" of Anti Financial Crime [14] [15]. It was founded in 1989 as an intergovernamental body that includes over 180 countries, with the goal to establish benchmarks that encourage the effective implementation of regulatory, operational and legal measures to combat risks associated with money laundering in the international financial system.

The Financial Action Task Force (FATF) plays a crucial role in shaping global anti-money laundering (AML) and counter-terrorist financing (CTF) policies. It develops Recommendations, known as the FATF Standards, which serve as an international framework for preventing financial crimes. These standards consists of 40 recommendations, divided in 7 groups, that encompass the whole landscape of monitoring, cooperation and shared responsabilites in the field of anti financial crime. These recommendations are regularly updated to address emerging threats, such as the misuse of virtual assets and cryptocurrencies, or cybercrime and dark web related crimes.

FATF conducts mutual evaluations of its member countries to assess their compliance with Anti-Money Laundering (AML) and Countering the Financing of Terrorism (CFT) regulations, offering guidance on necessary improvements. Countries that fail to meet FATF's standards may be placed on the "grey list," subject to increased monitoring, or the "blacklist," as high-risk jurisdictions facing countermeasures. This system of mutual recognition and shared standards establishes an effective framework of "soft law," encouraging countries to comply with FATF recommendations in order to maintain their standing in the global financial system and avoid reputational and economic consequences.

In addition to the policy development, FATF also fosters international cooperation by working closely with regional bodies, law enforcement agencies, and financial institutions to enhance transparency and disrupt illicit financial networks. Its influence extends beyond its membership, as many non-member countries align their regulations with FATF standards to maintain access to the global financial system.

1.3.2 European Regulatory Framework

The European Union has established a comprehensive regulatory framework to combat financial crime, particularly focusing on Anti-Money Laundering (AML). This framework is anchored by a series of Anti-Money Laundering Directives

[2] `https://www.fatf-gafi.org/en/home.html`

(AMLDs)[3] that impose stringent requirements on financial institutions, businesses, and designated non-financial sectors to implement customer due diligence (CDD), report suspicious transactions, and enhance transparency.

In response to the Financial Action Task Force's (FATF) recommendations, the EU introduced its initial AML Directive in 1991. This directive required member states to implement measures obliging financial institutions to identify customers, maintain transaction records, and report suspicious activities, laying the groundwork for a unified approach to AML across Europe. Over the following decades, the EU has adopted additional Directives to adapt to the evolving landscape of financial offenses. The latest directive, the Sixth Anti-Money Laundering Directive (6AMLD)[4], came into force on December 3, 2020, with a compliance deadline of June 3, 2021. It further strengthened enforcement by expanding criminal liability, increasing penalties, and harmonizing definitions of money laundering across EU member states, also defining a specific list of 20 predicate offences related to money laundering.

To enhance supervision and coordination among member states, the EU established the central Anti-Money Laundering Authority (AMLA) through Regulation (EU) 2024/1620, which entered into force on June 26, 2024. AMLA is tasked with overseeing compliance and facilitating intelligence-sharing among Financial Intelligence Units (FIUs) and the European Banking Authority (EBA) [16]. These measures collectively ensure a unified and proactive approach to preventing financial crime across the 27 EU member states.

1.3.3 Italian authorities for AML/CTF

The Italian anti–money laundering framework has evolved in accordance with international standards and European directives. Several legal provisions have been introduced to ensure compliance with supranational Directives and Recommendations: Decree Law 22 June 2007 n. 109; Legislative Decree 231/2007 (as amended by Legislative Decree 90/2017, transposing EU Directive 2015/849, and further aligned by Legislative Decree 125/2019, transposing EU Directive 2018/843); Decree Law 30 October 2019 n. 125; and Decree Law 26 October 2019 n. 124. Legislative Decree 231/2007, Italy's primary AML/CFT law, transposed EU Directive 2005/60/EC and introduced a risk-based approach, customer due-diligence and record-keeping obligations, suspicious-transaction reporting duties, and established the UIF as the national Financial Intelligence Unit. Its subsequent amendments strengthened

[3] Anti-Money Laundering Directives (EU).

[4] `https://www.europarl.europa.eu/legislative-train/theme-an-economy-that-works-for-people/file-6th-directive-on-amlcft-(amld6)`

beneficial-ownership transparency, extended the perimeter of obliged entities (e.g., cross-border service providers), imposed more stringent internal controls and training obligations, and expanded due-diligence duties to virtual-asset service providers and prepaid-card issuers [4]. The application and enforcement of these laws is distributed across both the public and private sectors.

Minister of Economic and Finance is the central governmental body responsible for formulating and implementing economic and financial policies in Italy. Within the MEF, the Financial Security Committee (FSC) plays a pivotal role in developing national strategies to prevent money laundering and terrorist financing. Established by Decree Law 369/2001 and governed by Legislative Decree 109/2007, the FSC coordinates efforts among various national authorities and ensures compliance with international sanctions, including the freezing of assets belonging to designated individuals and entities.

Figure 1.2: The main italian actors in AML/CTF efforts

Anti-Mafia Investigation Department (DIA) and **Special Currency Police Unit (NSPV)** are specialized law enforcment structures dedicated to

respectively combating organized crime and economic and financial violations. They are responsible of processing UIF reports related to their field of application, and in turn to inform UIF of relevant investigative results obtained in the same field.

The **Italian Financial Intelligence Unit (UIF)**, operating under the auspices of the Bank of Italy, serves as the central agency for collecting and analyzing information related to potential cases of money laundering and terrorist financing. It works in direct contact with LEAs and financial institutions to process and coordinate reports of suspicious activities and investigations on a national level. Financial institutions like Intesa Sanpaolo and other national banks directly transmit suspicious cases to UIF, primarily through reports of suspicious transactions (SOS), which in turn performs a more in-depth analysis of the case, integrating the report with information acquired from other entities under the coordination of UIF. Based on its findings, UIF produces reports for cases which are significant enough to be transmitted to NSPV and DIA for investigative inquiries. For this reason is important that institutions like Intesa Sanpaolo are able to research and employ advanced technological and intelligence tools like the ones proposed in this work. This assure that only significant and high-risk cases are escalated, reducing the number of false positives and improving efficiency across the investigative pipeline. Additionally, UIF serves a regulatory function by publishing guidelines, identifying trends and vulnerabilities in the financial system, and issuing recommendations to Italian obliged entities. To ensure compliance with international AML/CFT obligations, UIF also conducts inspections aimed at verifying adherence to reporting requirements, assessing communication protocols, and acquiring relevant data from financial and non-financial entities when necessary.

1.3.4 Obliged Entities

Aside from national public organizations and regulatory bodies, a wide range of public and private entities are classified as obliged entities under Italy and UE's anti-financial crime framework. These entities are legally required to cooperate with the Financial Intelligence Unit (UIF) and law enforcement agencies (LEAs) in preventing, detecting, and reporting financial crimes. This cooperation can take various forms, including monitoring customer activities for potential links to criminal actions and reporting suspicious behavior directly observed in operational settings—such as bank branches, where employees interact with customers in person.

The scope of obliged entities has significantly expanded in the recent years,

particularly with the Fifth European Directive 2018/843 [5]. Initially limited to banks and financial institutions, the designation now includes a broader range of professionals such as notaries, lawyers, certified public accountants, auditors, and audit firms. Additionally, non-financial operators involved in money-related activities—such as debt recovery agencies, companies providing custody and transportation of cash, securities, or valuables, and professional gold traders—are also subject to these obligations.

This mechanism is designed to eliminate blind spots in the financial system and ensure that those working closely with potential criminals are not incentivized to overlook or facilitate illicit activities. Failure to comply with these obligations can result in severe penalties, including administrative sanctions and criminal liability. By broadening the scope of obliged entities and reinforcing their legal responsibilities, the regulatory framework strengthens Italy's overall capacity to combat money laundering, terrorist financing, and other financial crimes.

1.4 Intesa Sanpaolo's positioning in the AML/CTF

Given its prominent effort in the landscape of Italian financial institutions, Intesa Sanpaolo plays an important role in AML/CFT operations, not only as a legally obliged entity but also as a key contributor of research and expertise to the AML/CFT community. As a well-established and deeply integrated institution, it provides valuable insights and know-how that support the broader fight against financial crime. ISP's primary obligations include reporting any suspicious or potentially criminal activities among its customers to the Financial Intelligence Unit (UIF). To comply with this obligation, the AML activities are grouped in the Compliance Area within a specialised Central Department, namely Anti Financial Crime Central Department, that acts as a pivotal unit in charge of the overall AML/CFT activities in the Intesa Sanpaolo Group.[13] Considering the Italian market, namely ISP Head Office directly enforces the AML/CFT provisions in the pertinent Business Units such as Banca dei Territori (BdT), Corporate and Investment Banking (CIB) and the Private banking pole. In this perspective, the transaction monitoring (TXM) and reporting of suspicious activities within Intesa Sanpaolo are structured through multiple processes involving various central organisational units, each responsible for one or more aspects of the overall process.

The pipeline is structured in four different levels, assigned to offices belonging to different directions, to ensure a broader point of view and mutual assurance

[5] `https://eur-lex.europa.eu/eli/dir/2018/843/oj/eng`

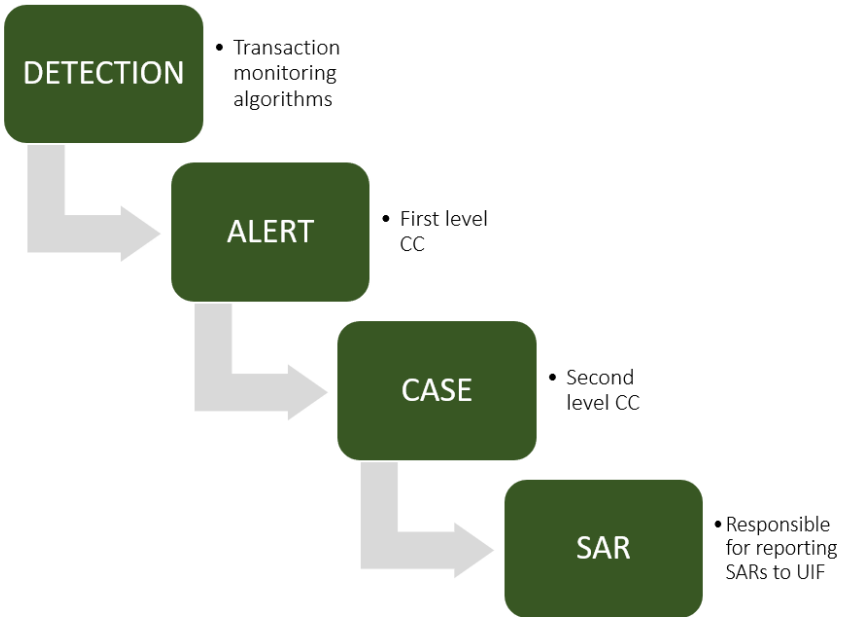


Figure 1.3: Operational pipeline from detection to SAR

of the quality of the overall process. In the first stage, the business activity of all bank's customers is analyzed at the lowest possible aggregation level, which consists of the transactions operated by customers in all its forms. This monitoring process involves different kind of payments, such as wire transfers, debit and credit card payments, and cash operations. This broad scope of monitoring assures that criminals cannot hope to go unnoticed by using a specific kind of payments methods. The output of this step, which is the one of main interest in the context of this thesis, are Detections. Detections are then sent to the first level Competence Center that has to refine them and discard irrelevant or wrong ones. The output of this filtering operation, which is called an Alert, is sent to the second level Competence Center, which again refines the input, possibly integrating it with other sources of information or linking together alerts at an higher level to produce a Case. Finally, relevant cases are sent to the last level of the pipeline which is responsible of reporting relevant ones to UIF in the form of Suspicious Activity Report, or SAR. By analyzing this kind of pipeline it is clear the importance of producing accurate detections already in the first level, with a particular focus on assuring low number of false positives, to limit the workload on downstream teams and ensure that investigative resources are focused on genuinely suspicious behaviors.

1.5 AML Transaction Monitoring

As stated previously, the term **Transaction Monitoring (TXM)** refers to the set of procedures and operations designed to continuously monitor transaction flows in order to detect potential money-laundering and terrorist-financing patterns. A significant portion of the transactions processed by financial institutions is conducted via wire transfers, and research conducted by leading law-enforcement and financial-intelligence agencies indicates that wire transfers constitute the primary method by which money mules channel illicit funds[6]. Since all proposed solutions in this thesis are based on data drawn from Intesa Sanpaolo's wire-transfer dataset, the following section provides a concise overview of the wire-transfer mechanism.

1.5.1 Wire Transfers

A wire transfer is an electronic method for moving funds between financial institutions or through specialized transfer agencies. The sender provides the recipient's details—name, account number, bank information, and purpose of the transfer—and pays any fees upfront. The sending institution then transmits payment instructions via a secure messaging system. Upon receipt, the beneficiary's bank credits its own reserves to the recipient's account, and the two banks settle the transaction later.

Wire transfers rely on three core components:

1. **Clearing System:** Reconciles and nets transactions before settlement, ensuring that obligations among participants are calculated accurately.

2. **Settlement System:** Executes the final exchange of funds. Settlement systems may operate as:

   - *Net Settlement Systems*, which aggregate and net payment obligations before settlement;
   - *Real-Time Gross Settlement (RTGS) Systems*, which settle each payment individually and immediately without netting;
   - *Real-Time Final Settlement Systems*, which combine RTGS with irrevocable, immediate settlement.

3. **Messaging:** Facilitates the secure exchange of transaction details between banks and the clearing/settlement infrastructure.

[6] `https://www.europol.europa.eu/crime-areas/forgery-of-money-and-means-of-payment/money-muling`

In many regions, banks are connected through a *centralized clearing and settlement system*, managed by a central bank or designated operator. This avoids the complexity of every bank maintaining direct bilateral links. In the Euro area, for instance, retail payments (typically up to €500 000) are processed through "ancillary systems" using net settlement in multiple daily cycles. The main platform for SEPA credit transfers is **STEP2**—a clearing and settlement mechanism operated by EBA Clearing [17]. STEP2 receives payment orders, calculates net positions among participating banks, and sends settlement requests to **TARGET2**. TARGET2, managed by the European Central Bank and national central banks, is an RTGS system that settles each payment individually and in real time. SEPA messages on STEP2 and TARGET2 follow the **ISO 20022** standard.

1.5.2 SEPA

Within the Single Euro Payments Area (**SEPA**), euro-denominated transactions are harmonized across participating EU and EEA countries under a common rulebook and technical standards. SEPA does not move funds itself; rather, it defines the specifications that payment service providers (PSPs) implement to exchange payment instructions. SEPA credit transfers and direct debits use ISO 20022 XML message formats leveraging IBAN and BIC identifiers to ensure consistent party identification. Governed by the European Payments Council (EPC), SEPA transactions are cleared and settled via TARGET2 or domestic clearing systems, enabling efficient, low-cost cross-border euro payments.

1.5.3 SWIFT

Outside of centralized regional networks, global transfers leverage the **SWIFT** (Society for Worldwide Interbank Financial Telecommunication) system. SWIFT also does not move funds itself; it provides a standardized, secure messaging system that connects banks worldwide. Each SWIFT message has a defined structure and a three-digit code (e.g., "MT 103") indicating its category, group, and specific purpose.

1.5.4 New Regulatory Expectation: From MT to MX (ISO 20022)

As payment methods evolve, regulators and market infrastructures are migrating from MT to **ISO 20022** XML-based messages (often called "MX" messages). The transition, which began in March 2023 and spans roughly two years, aims to:

- **Enhance Communication:** XML's structured, self-descriptive format improves interoperability across systems.

- **Provide Data-Rich Messages:** A single ISO 20022 message can include more comprehensive information—such as new roles (Ultimate Debtor, Ultimate Creditor)—reducing errors and facilitating compliance (e.g., AML checks, regulatory reporting).
- **Ensure Global Adoption:** Over 70 markets are planning or executing migration strategies, affirming ISO 20022's role as the emerging global standard for financial messaging.

MX message names encode their function and category similarly to MT codes, but take advantage of rich XML schemas to deliver structured, extensible data. This migration is a key step toward making cross-border and domestic payments more efficient, transparent, and interoperable.

1.6 TXM Guidelines and Challenges

Transaction Monitoring (TXM) is the first line of defense in the field of Anti-Financial Crime, as it enables financial institutions to detect illicit behaviors already at the transactions level. To support TXM, the UIF defined a list of 34 "anomaly indicators," which serve as rules and scenarios to guide analysts in identifying potential financial offenses and to harmonize the Italian TXM landscape. These indicators are grouped into three sections:

- **Section A (1–8)** focuses on the behavior and qualifying characteristics of the subject (e.g., refusal to provide required information, artificially complex ownership structures, involvement of high-level politically exposed persons).
- **Section B (9–32)** addresses the features and configuration of the operations themselves, often in relation to specific sectors (e.g., repeated or fragmented high-value cash or crypto-asset transfers, suspicious real-estate or precious-metal dealings and money-transfer activity).
- **Section C (33–34)** covers activities potentially linked to terrorist financing and to the proliferation of weapons of mass destruction (e.g., transfers to counterparties in high-risk or embargoed jurisdictions, transactions involving entities with inadequate AML controls).

Despite relying on these fundamental indicators, detecting all money-laundering and financial-crime patterns at the transaction level remains a challenging, open problem for financial institutions. There are many factors that make this challenge difficult. First, although the total volume of laundered funds worldwide is huge, money laundering still represents only a small fraction of overall banking activity and thanks to money launderers efforts it can easily go undetected without proper

monitoring structures and methods. Second, the sheer scale of data itself represents a problem: Intesa Sanpaolo, for instance, processes nearly 60 million wire transfers each month across a wide variety of customers and sectors, and must accurately flag the few operations that may be illicit in as little time as possible. Third, money laundering is all but a static procedure: every year, the rise of new technologies (such as cryptocurrencies) introduces new, less traditional laundering channels, which are often difficult to trace thanks to their relative anonymity. Stacking all these challenges, performing Transaction Monitoring effectively becomes a massive undertaking for analysts, who must still heavily rely on human expertise and weakily automated frameworks. To give an idea of the challenges posed by this task, it can be observed that, despite deep human domain knowledge, advanced technological tools and regulatory efforts at supranational level, at the current state of TXM pipelines and operations, only a small fraction of produced SOS actually represented real criminal operations.

Another key consideration is that the vast volume of data financial institutions must process is inherently unstructured (as explained in Chapter 2) and does not directly expose properties useful for detecting money-laundering patterns, such as temporal sequences, relational ties, or logical connections among transaction counterparties. For these reasons, it is paramount to develop robust, automated frameworks that can extract and leverage the latent structure within transactional data. In particular, **Network Analysis (NA)** offers a powerful paradigm: by representing transactions as graphs—where nodes denote entities and edges represent financial flows—techniques such as community detection, centrality measures, and graph-based anomaly detection can uncover hidden relationships and identify atypical interaction patterns [18] [19] [20]. Furthermore, the incorporation of **Artificial Intelligence (AI)**, including machine learning and deep learning models, enables the capture of underlying, complex patterns and evolving schemes that rule-based or manual methods may overlook, thereby significantly enhancing the predictive accuracy and adaptive capacity of TXM systems.

This thesis presents two novel, proprietary algorithms grounded in NA/AI methodologies. The first algorithm provides analysts with a systematic framework for detecting money-muling patterns in the transactional behaviour of a given target under the Bank's oversight scope. The second algorithm is a graph-based recurrent neural network that both embeds and forecasts macro-economic trends across the millions of transactions processed by the Bank each month, thereby enabling automatic flagging of suspicious money flows at a high level of aggregation—potentially indicative of money laundering, terrorist financing or sanctions-evasion. Both approaches achieve performance on par, or better, with state-of-the-art solutions and are already deployed within the Anti-Financial Crime (AFC) department of Intesa Sanpaolo.

Chapter 2

Networks and Network Analysis

This chapter provides an overview of fundamental definitions in the field of complex networks and network analysis as a scientific discipline. Initially, general concepts related to network definitions, representations, and classifications are introduced, followed by an examination of both local and global properties relevant to this study. Subsequently, the focus shifts to applications within the financial sector, particularly in the context of Anti-Money Laundering (AML) and Countering of Financing of Terrorism (CFT).

2.1 Network Theory

A network consists of a set of entities, usually called nodes, along with a set of connections, or edges, of a specified type that link them. Edges connect nodes, to form paths in the networks that indirectly link nodes that are not directly tied. The pattern of ties in a network yields a particular structure, and nodes occupy positions within this structure. Much of the theoretical wealth of network analysis consists of characterizing network structures (e.g., small-worldness) and node positions (e.g., centrality) and relating these to group and node outcomes.

Network theory is a subset of graph theory, which was developed to answer questions about connectivity and optimization of systems that can be represented by nodes and edges. This theoretical framework has been used to answer questions regarding a vast range of different topics, as for example human social networks, biology related applications like genes expression, and finally financial related problems like the one this work focuses on [21].

2.1.1 Historical Background

Graph theory is one of the few mathematical fields with a well-defined origin, dating back to 1736, when Leonhard Euler introduced a mathematical formulation of the Königsberg Bridge Problem [22]. The challenge was to determine whether it was possible to cross all the bridges in the city of Königsberg (now Kaliningrad, Russia) exactly once and return to the starting point. By representing the problem as a graph, Euler demonstrated that such a path was impossible, introducing key concepts like nodes (vertices) and edges as fundamental components of a network. His work laid the foundation for graph theory and topology, influencing modern studies in network analysis and connectivity.

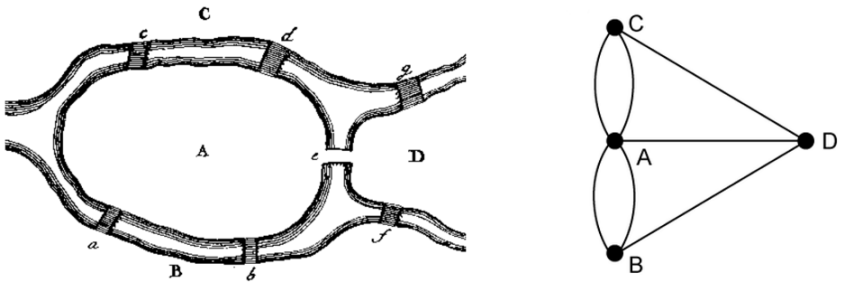


Figure 2.1: Konigsberg Bridges problem and Euler graph representation

Despite scattered contributions in the 18th and 19th centuries from figures like Cauchy, Kirchhoff, Hamilton, and Poincaré, a formal and systematic study emerged only in 1936 with Dénes "Kőnig's Theorie der endlichen und unendlichen Graphen" [23]. This book provided the first comprehensive treatment of graph theory, covering fundamental concepts such as graph connectivity, planarity, and matching theory, forming the basis for modern graph research.

In parallel, sociologists in the early 20th century began applying graph theory to analyze social structures. Jacob Moreno [24] pioneered sociometry, introducing sociograms to map relationships within groups. His work explored how interpersonal connections influence group dynamics, an approach that later evolved into Social Network Analysis (SNA). Expanding on this, Mark Granovetter [25] introduced the concept of weak ties, demonstrating their importance in the diffusion of information and job opportunities. Around the same time, Stanley [26] conducted his small-world experiment, which provided empirical evidence for the "six degrees of separation" phenomenon.

A major breakthrough came in the late 1950s, when Paul Erdős and Alfréd Rényi introduced random graph theory, providing a probabilistic framework for understanding how networks form and evolve [27]. Their ER model showed that

random connections between nodes could lead to the emergence of large-scale network structures, laying the groundwork for network science. However, by the late 1990s, researchers realized its limitations in explaining real-world networks, such as the Internet, biological systems, and financial networks.

This led to the development of new models, notably the Watts-Strogatz small-world model [28], which demonstrated that real-world networks exhibit short path lengths and high clustering—explaining the "small-world" property observed in social, biological, and technological networks. Around the same time, Barabási and Albert (1999) introduced the scale-free network model, showing that many real-world networks follow a power-law degree distribution, where a few highly connected nodes (hubs) dominate the structure [29].

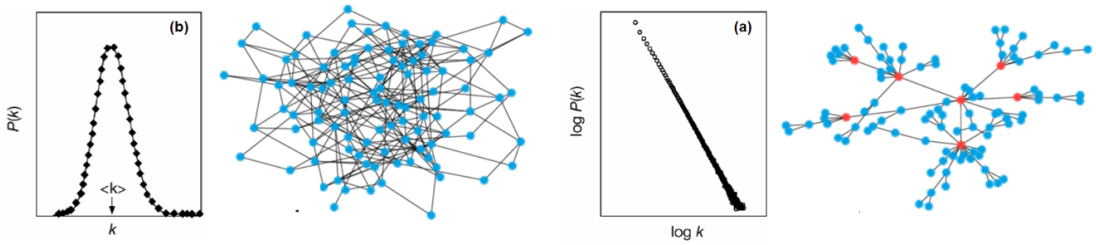


Figure 2.2: Random network and its nodes degrees distribution (a) against Scale Free network (b)

Since then, network science has become a highly interdisciplinary field, integrating contributions from physics, mathematics, computer science, biology, economics, and sociology. Today, network analysis plays a crucial role in studying complex systems, including financial networks and, and can be employed to map and characterize money laundering offences that this work aims to detect (see [30]).

2.1.2 Networks Representation

Once a network is defined, it becomes necessary to establish a mathematical formulation that captures its key properties in a functional and theoretically robust manner. In most cases, this is achieved by representing networks as graphs—mathematical structures composed of nodes and edges. Nodes encapsulate relevant information about distinct entities within the network, while edges define various types of logical connections between pairs of nodes. By leveraging the well-established definitions, properties, and algorithms of graph theory, this representation naturally accommodates a broad range of network characteristics, including not only nodes and edges in their simplest form but also paths, structural patterns, connected and disconnected components, and analytical measures such as centrality, betweenness, and degree, among other key network attributes.

While embedding information in nodes is relatively straightforward, representing edges presents a more complex challenge. In complex networks, the number of connections often greatly exceeds the number of entities, creating the need for an efficient and scalable method to store connection data. Various representations exist for this purpose, each with its own advantages and trade-offs. Edge lists provide a compact and memory-light format suitable for algorithms that iterate over all edges. However, they carry a linear O(E) cost to access a specified edge amongst the edges set E. Adjacency lists, on the other hand, store edges on a per-node basis, typically using arrays or linked lists, making them efficient for neighbor lookups. Adjacency matrices, though more memory-intensive—particularly for sparse networks—enable O(1) access to edges, making them advantageous for operations requiring fast indexing. The choice of representation depends on factors such as network size, density, computational requirements, and memory constraints.

2.1.3 Network Types

Graphs, and consequently networks, can be classified into different types based on the properties of their nodes and edges [21]. The first distinction concerns the presence of self-loops and parallel edges. A self-loop is an edge that connects a node to itself, while parallel edges are multiple edges linking the same pair of nodes. A graph that contains neither self-loops nor parallel edges is classified as a Simple Graph. However, this category is relatively simplistic and is not well suited to model real-world complex networks. If a graph includes self-loops or parallel edges, it is referred to as a Multigraph.

The second major distinction relates to the directionality of the relationships between entities in the network. If connections have a well-defined direction, the graph is classified as a Directed Graph, meaning edges can be traversed only in the specified direction. Conversely, if directionality is either unnecessary or undefined within the original network, the graph is considered an Undirected Graph.

Finally, edges can be further categorized based on their weight, which represents the significance or strength of the connection. In a Weighted Graph, edges carry a numerical value—such as cost, capacity, or distance—associated with the relationship they represent. If all edges are treated as equally significant regardless of their properties, the graph is classified as an Unweighted Graph.

Graph Representation in AML

In the context of Anti-Money Laundering (AML), this study models follows [31] notation, embedding nodes as financial entities—such as individuals, companies, and financial institutions—identified through unique identifiers like IBAN codes (for bank accounts) or BIC codes (for financial institutions). Transactions between these entities are represented as edges connecting the graph nodes, with attributes such as transaction amount and timestamp defining their characteristics.

It is important to note that this representation is not universal. Several studies take alternative approaches, one of the most common being the direct embedding of transactions as nodes (see [30]), while edges represent other types of relationships between transactions, such as temporal correlations or statistical dependencies. Different modeling choices reflect varying analytical needs and research perspectives within the number of possible AML applications.

2.2 Network Analysis in AML/CFT

In order to spot suspicious activities, the unit of analysis in a transaction monitoring framework (TXM) may focus on individual transactions, if the condition being assessed pertains to specific attributes of them. Alternatively, it may analyze groups of transactions when the condition to be investigated requires identifying patterns across multiple transactions. In the latter case, this approach can naturally be framed as a "network" problem, where the distinguishing features of illicit behaviors emerge only by analyzing and extracting meaningful patterns from the structure of the network. In such solutions the network is modeled as a graph, where nodes represents the entities of interest and edges can represent any logical connection between two or more of them. This data representation can be extremely versatile, as it allows to represent and visualize the same set of data in different ways depending on the choice of node and edges. The approach of combining different features extracted from the network can be applied throughout the financial intelligence supply chain: from the financial activity data used in the TXM stage to the processing of FIU reports by law enforcement agencies. This work primarily focuses on the TXM stage, exploring various potential enhancements achievable through Network Analysis.

Works applying Network Analysis to AML/CTF problems have become increasingly common in recent years. Authors of [32] estimate a 61% annual increase in related scientific publications since 2010. This growing interest underscores the potential of network-based tools but has also led to a fragmented landscape of approaches, where no clear standard has emerged in terms of techniques and validation methods. The rising importance of Network Analysis, along with the lack of cohesion between institutions, has been recognized by Europol[1], which includes network analysis automation, development, and integration in its 2024-26 programming document for AML/CFT pipelines.

Another major challenge in the field is the limited collaboration between research institutions and frontline entities, such as Banks, Financial Intelligence Units, and Law Enforcement Agencies. While recent years have seen improvements in this

[1]Europol Programming Document 2024-26.

regard, a widespread and structured cooperation that effectively guides research directions and consolidates findings remains lacking. This gap is particularly evident in data sharing. Most real-world money laundering cases involve proprietary financial data that institutions cannot or do not wish to share, often due to strict regulatory constraints. User transaction data is protected by GDPR and similar laws, making it difficult for researchers to access real-world data for model evaluation. As a result, authors of [32] report that only half of the analyzed models have been trained or tested on real proprietary data, while the other half rely on publicly available or synthetic datasets, that are often lacking or simply too old to keep pace with the fast changing landscape of possible ML patterns and offenses.

Despite the above difficulties, a significant number of innovative approaches has been proposed. Analyzing existing literature on the subject, it is possible to draw a general picture of the areas of innovations, the methods utilized to achieve results and the current limitations of the state of the art solutions.

2.2.1 New Directions of Research

There are many possible objectives that can be reached by applying Network Analysis principles in the Transaction Monitoring and AML/CTF problem. In the following, the most notable ones that emerge from existing literature are listed, accompanied by a brief description of motivations and goals of each analysis.

The most common objective in the observed publications is client classification. This process involves analyzing payments made over a period of time along with a client's objective and subjective characteristics to construct a representation of their typical behavior. This in turn enables the application of supervised or unsupervised classification and clustering methods to distinguish between legitimate and potentially criminal users within the network. This task essentially translates into an Anomaly Detection problem, a well-established and robust area in Machine Learning and AI research. Beyond identifying anomalous users, this behavioral representation also facilitates the comparison of a client's activity over different time periods, allowing for the detection of sudden transactional shifts that may indicate the onset of suspicious operations. This capability is particularly valuable in the context of AML/CTF efforts, as it is fairly common for money mules and launderers to exploit seemingly "dormant" accounts—those with little to no prior activity in the databases—to carry out criminal operations. By identifying sudden shifts in operativity, such as an unexpected increase in transaction volume or frequency, these methods can help detect and flag potentially illicit activities at an early stage.

Shifting focus from clients to the transactions themselves, another key area of innovation is the detection of suspicious "flows" of transactions. The definitions of what constitutes a transaction flow, as well as what makes a flow suspicious, are not

universal and vary across different studies, further highlighting the fragmentation of the research landscape in this field. Some authors [33] [34] focus solely on the temporal properties of transactions, linking two or more transactions if they occur within the same time window. The range of this window is typically determined either by optimizing the proposed models on test data or by leveraging prior sector-specific knowledge from the AML/CTF domain. Generally, these time windows range from a few hours to several days. Other studies [35] [36] also take into account the overall monetary value that characterizes a transaction flow, filtering out cases where transactions are temporally related but involve amounts that do not suggest real correlation, or that are simply too small to suggest any meaningful criminal intent. In more complex formulations, some authors [37] [38] [34] define metrics to assess statistical correlations between transactions within the network. These correlations may be based on time and amount parameters or may work alongside them to provide additional insights into transactional relationships. These kinds of models, although still not optimal, enable researchers to expand their set of mathematical tools by incorporating probabilities and statistical concepts. This, in turn, allows for the development of models that are better suited to capturing and mapping complex behaviors within the network—behaviors that closely reflect real-world phenomena, such as those under investigation in this study. By identifying suspicious flows, these models enable investigators to isolate the involved nodes and uncover hidden structures within money laundering schemes, ultimately enhancing the effectiveness of AML/CTF operations.

Finally, another promising area of research involves the visualization of financial-related networks. While not inherently tied to efforts in combating money laundering, this challenge is part of the broader field of complex network analysis. When the network under investigation consists of millions of nodes and potentially billions of edges, effectively conveying all relevant information in a human-readable manner becomes a highly non-trivial task.

At the same time, for models to be practical and usable by real-world AML/CTF professionals, it is essential to develop methods that allow for the isolation and exploration of different perspectives within the same network structure. Given that the sheer volume of elements far exceeds the feasible range for direct visualization, the key lies in aggregating data in meaningful and flexible ways.

The approaches found in the literature vary widely, with aggregation methods based on factors such as geographical location, keywords in remittance information, temporal patterns, and other domain-specific attributes. Developing effective visualization techniques not only enhances interpretability but also aids investigators in identifying hidden patterns, tracking illicit flows, and generating actionable intelligence more efficiently.

The first algorithm proposed in this thesis (see Chapter 3) addresses the three key objectives outlined above within a unified framework, equipping analysts with client

classification, transaction-flow detection, and an innovative time-based visualization tool to enhance human interpretability.

2.2.2 Characterization Methods

Aside from the different objectives of analysis, the proposed solutions can also be categorized based on the methods used to achieve results. The two main methodological families observed in the existing literature are Supervised methods and Unsupervised methods.

Supervised methods refer to models that learn patterns or data representations using labeled data and generalize them to make predictions on new, unseen data. Common supervised techniques include classification (e.g., fraud detection, where transactions are labeled as fraudulent or legitimate) and regression (e.g., predicting transaction amounts). Unsupervised methods, on the other hand, work with datasets where no explicit labels are provided. These models identify hidden structures or patterns within the data without prior knowledge of expected outcomes. Common unsupervised techniques include clustering (e.g., grouping similar customers based on transaction behavior), features auto-encoding and anomaly detection. Finally, models that leverage both labeled and unlabeled data, or generate labels even when not explicitly given, fall into the category of Hybrid or Semi-supervised methods.

A key observation when analyzing existing literature through this framework is that the vast majority of studies rely on unsupervised methods [32]. While these might not seem the most intuitive choice for some of the tasks mentioned earlier, the reasoning behind this preference lies in the already discussed data availability challenge. Money laundering is an inherently complex and elusive phenomenon, where a significant portion of illicit activity remains undetected [11]. Furthermore, the landscape of financial crime is constantly evolving, with new schemes and typologies emerging each year. Additionally, money laundering represents only a small fraction of overall financial activity, resulting in highly imbalanced datasets with very few labeled examples of illicit behavior.

For these reasons, constructing a reliable labeled dataset—where all money laundering attempts are correctly identified as fraudulent, and all legitimate transactions are accurately classified as genuine—is an immense challenge. As a result, unsupervised methods often become the most practical and viable approach. Moreover, even when labeled datasets are available, such procedures would be more accurately described as semi-supervised rather than fully supervised. This is due to the fact that, in real-world scenarios, many money laundering activities go undetected by authorities, meaning that datasets labeled as "genuine" may, in reality, contain unidentified instances of financial crime.

2.2.3 Related Works

Amongst all existing literature on Network Analysis applied to AML/CFT, some works are notably interesting for the scope of this study. They mainly concern flow chains detection, anomaly detection and visualization techniques. Several innovative systems and frameworks have been proposed for detecting money laundering through network analysis and visualization. Starting from their previous work Money Laundering Detection System (MLDS), a tool that performs data mining on transactional databases to extract frequent patterns and visualize them , Dreżewski, Sepielak, and Filipkowski (2015), propose [39] by incorporating Social Network Analysis (SNA) algorithms to analyze networks of bank account holders and entities from the National Court Register. Their work aims to classify entities in the network leveraging a set of known graph metrics, assigning roles based on pre determined metrics interval. Moreover, they also make an attempt at entity resolution, that is the process of recognizing same entities if they appear more than once in the network, by computing cosine similarities on the obtained metrics vectors.

Zhou et al (2017). introduced HOSPLOC [40], a local clustering framework for modeling higher-order network structures. HOSPLOC searches fo "structure-rich" higher-order subgraph that are scarcely connected to the rest of the network, suggesting a possibly suspicious behavior. The usage of higher-order representation networks, where original networks entities are aggregated based on defined heuristics is particularly interesting to obtain a more refined view of the huge original dataset, and is reproposed in FaSTMaN [34], a framework for constructing temporal graphs of sequential transactions proposed by Tariq and Hassani (2024), detecting closely related transaction sequences, and forming communities of flows. This works also stands out for its temporal approach to the transaction networks, introducing tools to robustly model temporal dependencies between nodes in order to validate found flows. FlowScope [41], developed by Li et al. (2020), marks a shift in the nature of the densest subgraph of interest: instead of just focusing on the densest graphs in terms of entities, it also take into account the cumulative amount of money that flows across those subgraphs. It models money laundering detection by identifying the densest multi-step flow in bank transaction graphs and offers theoretical guarantees on near-optimal dense flow detection, defining a limit on the amount of money that criminals can move without being detected. Starnini et al. (2021) [33] focused on detecting suspicious patterns inside the transaction network, defining two "smurf-like" motifs that criminal can use to divide large sums into smaller transactions—proposing a pipeline to identify suspicious subgraphs. Finally, Ovelgönne, Kang, Sawant, and Subrahmanian introduced the concept of Covertness Centrality (CC) [42] as a measure of an actor's ability to remain hidden within a network. It defines the similarity between actors based on the variance of centrality measures and proposes two "commonness" metrics to quantify how

similar an actor is to others in the network, demonstrating optimal concealment capability for nodes associated with max values

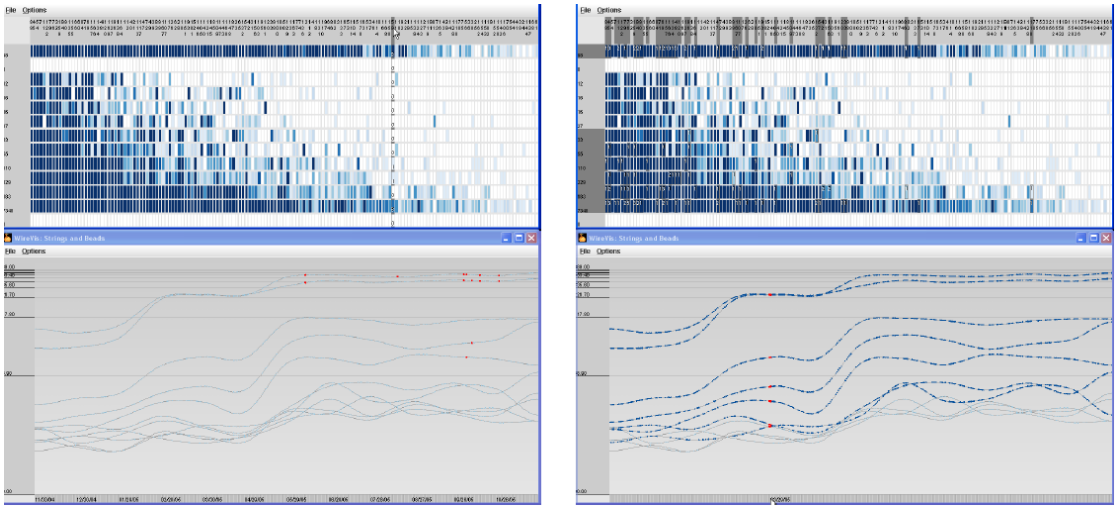


Figure 2.3: WireVis transactions visualization, heatmap and timeline

For the visualization tools, Chang et al. (2007) developed WireVis [43], in collaboration with Bank of America, to monitor wire transactions using coordinated visualizations based on keywords in the remittance info of wire transfers. This system, although quite old, has been really inspirational for this work, particularly for the focus it has on visualizing the temporal dimension of the network through a timeline, and the combination of different visualization tools to convey different aspects of the case in analysis. Furthermore, Cheong et al. (2010) proposed an event-based approach for money laundering data analysis and visualization in [44], leveraging a structured database to store crime records and detect clusters, suspicion degrees, and associations by iterating on node neighbors and computing subjective and objective metrics to assess the risk potential of the nodes

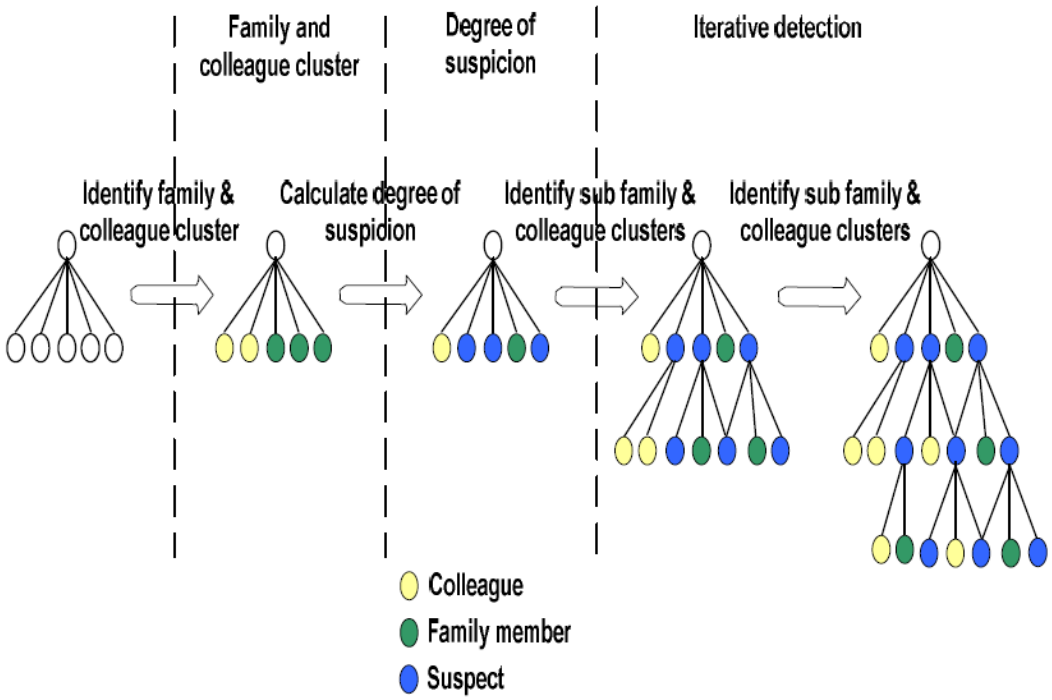


Figure 2.4: Suspicious network neighborhood generation, [44]

Chapter 3

TRACED Algorithm - Temporal Relational Analysis for Criminal Entities Detection

The first major contribution of this work is the development of TRACED: an algorithm designed to identify moneymuling-like transaction patterns associated with a given financial target managed by the bank.

Two key innovations are proposed in this section. First, TRACED automates large parts of the process of detecting complex money-laundering schemes, particularly those involving money mules, by replacing time-consuming, rule-based manual analyses with a fully automated, machine-learning–driven pipeline that ingests raw banking transaction records for any target—be it an individual, a company, or any other organization—and rapidly isolates money mule-style transactions flows. Second, TRACED introduces an innovative user interface (UI) to enhance the interpretability of its results. Its novel visualization tool provides a timeline-based representation of transactional flows, combining the strengths of both relational and temporal information acquired during the analysis. This visual framework allows investigators to accelerate the review of suspect targets and supports seamless export of annotated timelines and pattern summaries directly into intelligence reports—bridging the gap between algorithmic detection and actionable insight.

The following sections highlight the key components of TRACED design, its main outputs and an evaluation of its performance on a set of real-world money mules accounts taken from Intesa Sanpaolo's databases.

3.1 TRACED's Design and Workflow

TRACED's pipeline is made of six main steps (see 3.1): the first step is retrieving relevant transactional data corresponding to the target's activities within a specified period of interest (Sec. 3.2). This data is then structured into a meaningful and analyzable format, specifically a graph representation, to facilitate the extraction of relevant, relational and temporal based features for classifying each transaction (Sec. 3.3). The algorithm then leverages a combination of statistical methods, machine-learning strategies and domain-specific knowledge to analyze the data and produce three main outputs: a visualization of the target cumulative balance (Sec. 3.4), annotated with transactions values and circuit (SEPA/SWIFT/cash deposit-withdrawal); an innovative timeline representation of the transactions operativity of the target (Sec. 3.5), where transactions are aggregated in "flows" of related ones using a novel algorithm named TRACED-FaSTMaN, based on findings proposed in [34], and finally a target-level risk score (Sec. 3.6) generated with a topology-agnostic community detection algorithm inspired by [34] and a linear regression model applied to predictions regarding the likelihood of suspicious activity for each transaction.

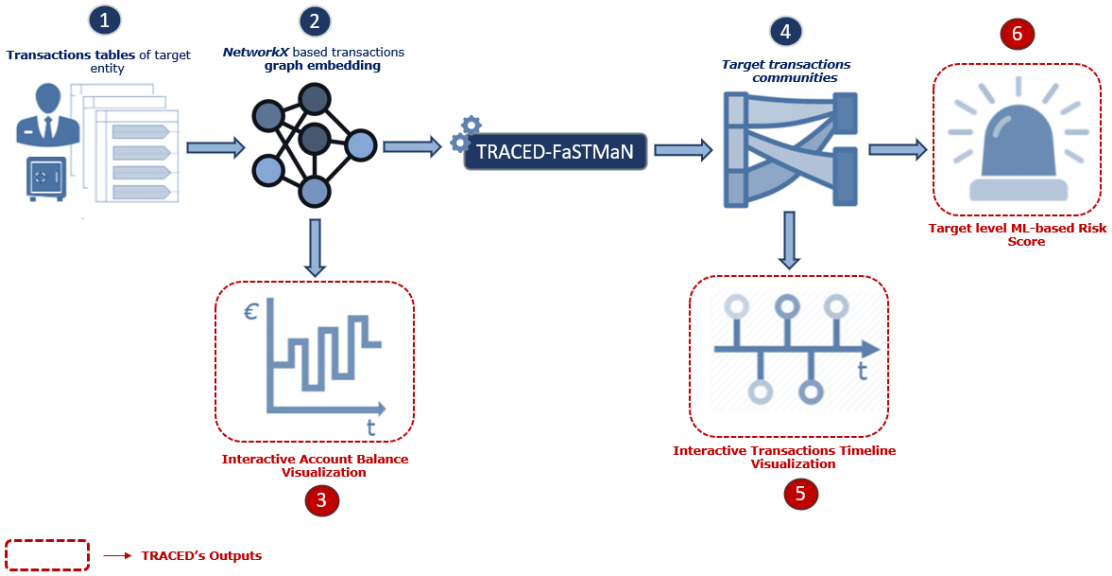


Figure 3.1: Operational pipeline of TRACED: (1) retrieve target transactions; (2) embed data as a graph; (3) build cumulative balance; (4) isolate suspicious flows via TRACED-FastMaN; (5) generate interactive timeline; (6) compute regression-based risk score.

3.2 Transactions Operativity Retrieval in TXM Pipelines

Although largely outside of the primary scope of the architecture proposed, it is essential to provide some context regarding the retrieval of transactional data for the selected targets, as it serves as the foundation for the subsequent stages of the algorithm. In this research, data extraction has been conducted using the data mining infrastructure of the AFC/MAS office of Intesa Sanpaolo. These extraction routines are implemented in SAS.

3.2.1 SAS (Statistical Analysis System)

SAS[1] is a powerful software suite used for advanced analytics, data management, business intelligence, and predictive modeling. Developed by SAS Institute, it is widely employed across industries such as finance, healthcare, and government for tasks including statistical analysis, fraud detection, and risk assessment. It provides both a comprehensive programming language and a graphical user interface, enabling users to manipulate large datasets, apply machine learning techniques, and generate insightful visualizations. Its capability to efficiently handle vast amounts of structured and unstructured data makes it a largely diffused tool in the world of data analysis and management.

The extraction routines facilitate a standardized and user-friendly procedure for retrieving operational data related to various payment circuits, such as wire transfers, debit and credit card transactions, and cash deposits or withdrawals. This data forms the fundamental core of the transactional network on which subsequent analyses are performed.

3.2.2 Workflows Currently in Use

Data extracted through SAS is typically output as large Excel spreadsheets, where each row represents a transaction conducted within the selected period, described by a huge number (in the range of almost hundred) of numerical and categorical features. While these files contain valuable information, they also present several limitations that make direct analysis challenging. The data can be noisy, potentially containing duplicated transactions, and lacks explicit relational properties, such as links between related transactions, temporal dependencies, or value-based correlations.

At the current state of most Anti-Money Laundering (AML) applications, workflows based on SAS output heavily rely on manual analysis of these large

[1] `https://www.sas.com`

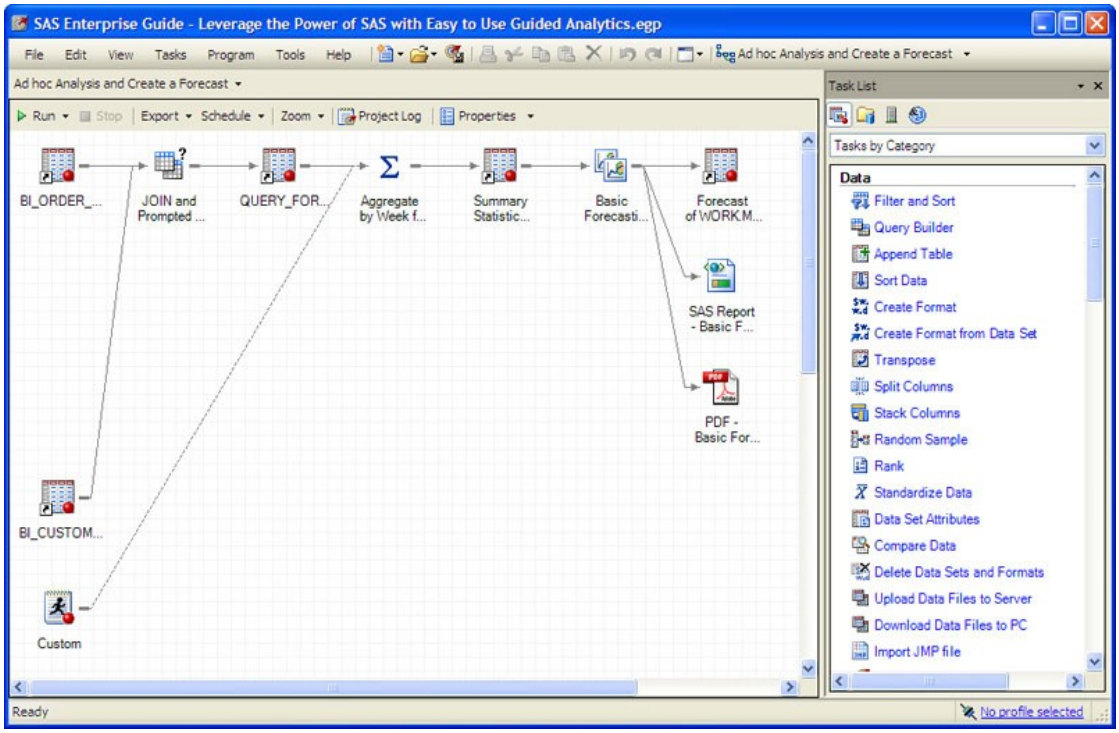


Figure 3.2: Example of SAS visual interactive queries system

transaction tables to identify potentially suspicious activities. Investigators must sift through extensive datasets, searching for anomalies, inconsistencies, or transaction patterns indicative of illicit financial behavior. This process can be really time-consuming, requiring investigators to recognize meaningful connections within raw data manually. Once suspicious activity is identified, findings are compiled into a dossier, which progresses through different stages of the AML/Counter-Terrorism Financing (CTF) operational chain, ultimately leading to reporting to regulatory authorities or legal proceedings.

While this manual approach is functional, it is far from optimal. It demands a significant workload and carries the inherent risk of overlooking complex, hidden patterns due to human limitations. The TRACED algorithm aims to address these challenges by providing investigators with an advanced analytical tool capable of automating the initial stag of the investigation. By systematically identifying and flagging potentially suspicious transaction patterns within the unordered dataset, the algorithm significantly reduces the time required for manual screening. This allows investigators to focus on refining the results, applying their domain expertise to interpret flagged transactions effectively, and ultimately producing high-quality intelligence reports.

3.3 TRACED's Transactions Graph Representation

Starting from transactions extracted through SAS queries, the initial task of the TRACED algorithm is to construct a meaningful network representation of these transactions. As discussed in Chapter 2, there is no universally established encoding of features into nodes and edges in the literature. The most common approaches include: (i) representing financial entities (e.g., IBANs, bank coordinates, or personal information) as nodes, with transactions forming the edges that connect them, or (ii) encoding transaction details directly into nodes while utilizing edges to map higher-level relationships between transactions, such as temporal, statistical, or logical connections.

TRACED leverages both representations to combine their strengths and capture different layers of information. In the data loading stage, the chosen approach is to encode customer-related information in nodes, ensuring that entities involved in transactions are clearly mapped. The algorithm provides users with the flexibility to select a specific field of interest from the database as the node key. This design choice provides a great degree of flexibility to the algorithm, allowing analysts to perform investigations at different aggregation levels by simply modifying the selected key while maintaining the same underlying software architecture.

Once the algorithm has created the graph-based representation of the original transactions data, it can use it to produce innovative and functional outputs for different doownstream tasks, that are explained in detail in the following section.

NetworkX Library for TRACED: To create the graph data-structures necessary for TRACED, the NetworkX[2] library is chosen. NetworkX is a Python package designed for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. By providing data structures for representing both undirected and directed graphs, as well as multigraphs, NetworkX allows researchers to model real-world systems—ranging from social networks to biological pathways—using a flexible, dictionary-based approach. Its rich set of built-in algorithms includes routines for computing shortest paths, clustering coefficients, network centrality measures, and community detection, enabling comprehensive topological analysis without the need to reimplement core graph algorithms. Furthermore, NetworkX's seamless integration with the wider Python scientific ecosystem (e.g., NumPy, SciPy, Matplotlib) facilitates data preprocessing, numerical computation, and high-quality visualization necessary for the different outputs of TRACED.

[2] `https://networkx.org/documentation/stable/reference/index.html`

3.4 TRACED Output/1: Balance Account Visualization

After the transaction data from an Excel spreadsheet have been accurately stored in a NetworkX Graph object, TRACED can produce its first output: the target's account balance visualization tool. This plot, and all the subsequent ones, are written using functions from Bokeh library, which is briefly described in the following paragraph, before delving into more details about the proposed visualizations.

UI Package for TRACED: Bokeh Bokeh[3] is an open-source Python library designed for creating interactive, browser-based visualizations. It leverages a high-level API that seamlessly translates Python code into dynamic HTML and JavaScript (via BokehJS), allowing researchers to build rich plots—such as scatter plots, line graphs, and heatmaps—with features like zooming, panning, and hover tooltips. Bokeh integrates smoothly with common data-science tools (e.g., NumPy, Pandas, and Jupyter Notebooks), enabling real-time data streaming and server-driven updates for dashboards. Consequently, Bokeh is widely used in academic and industrial contexts to prototype and deploy interactive visual analytics without requiring deep expertise in web development.

3.4.1 Balance Account Plot

In the account balance plot, the amounts of all transactions associated with the target during the analysis period are displayed over time, with colors denoting the transaction type or channel (e.g., SEPA, SWIFT, or cash deposit/withdrawal). This representation provides the analyst with an immediate quantitative overview of the target's activity trend and volume, thereby facilitating the rapid detection of potentially suspicious patterns. For instance, Fig. 3.3 compares two operational profiles: the first corresponds to an account linked to a wholesale dealer, which appears in the databases solely as a recipient of payments from the target; the second is drawn from adocumented financial fraud cause (specifically, money muling) extracted from ISP's closed-cases archives. The characteristic rapid, almost square-wave pattern of the fraudulent operations is readily discernible, in stark contrast to the legitimate accumulation observed in the wholesale-dealer account. Furthermore, employing distinct colours enables analysts to identify suspicious patterns related to transaction types, beyond mere volume or timing. For example, several case studies have demonstrated that cash deposits and withdrawals are commonly utilized in money-mule operations to facilitate rapid placement and layering of illicit funds [45].

[3] `https://docs.bokeh.org/en/latest/`

Likewise, SWIFT transfers directed toward jurisdictions flagged by the Financial Action Task Force as high-risk for money laundering frequently serve as red flags for potential transactions to non-EU countries of concern [46].

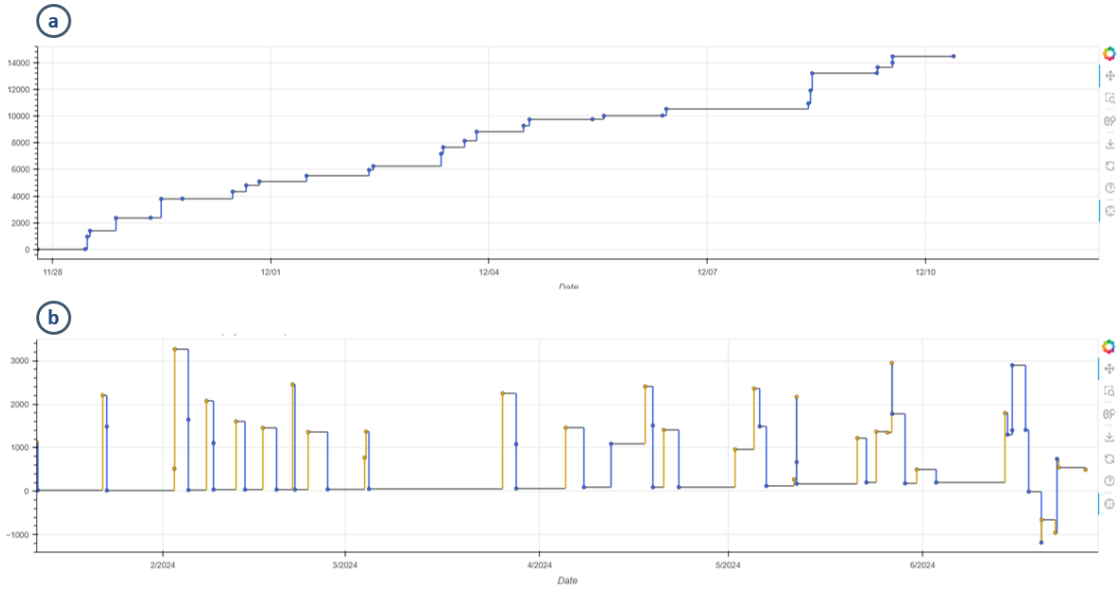


Figure 3.3: Comparison of two operational profiles: (1) a wholesale-dealer account, which only collects payments from the target; (2) a known money-mule account from the bank's closed-case database.

3.5 TRACED Output/2: Timeline Visualization

TRACED's second output offers an innovative visualization that allows to capture both temporal and relational dimensions of a target entity. Conventional network-centric approaches—such as those based on NetworkX's plotting functions (see NetworkX[4] or GraphViz[5])—primarily emphasize the spatial arrangement of nodes and edges, providing many ready-to-use layouts taken from academic literature. However, as graph size increases, these layouts rapidly become cluttered and none of them can convey readily accessible information about the temporal distribution of transactions or edges more generally.

Within the TRACED framework, the algorithm first constructs the complete

[4] `https://networkx.org/documentation/stable/reference/drawing.html`

[5] `https://graphviz.org/docs/layouts/`

transaction graph and then computes the 1-hop enclosing neighborhood (the so-called egonet of the node) for the target node. All transactions involving the target are subsequently grouped by counterpart and mapped onto a temporal axis. This representation enables analysts to observe both the relational aspect—identifying which entities interact with the target—and the temporal aspect—revealing the timing and sequence of those interactions. By delineating the "shape" of relationships over time, TRACED facilitates the detection of patterns (e.g., bursts of activity with particular counterparts) that would remain obscured in purely spatial network depictions.

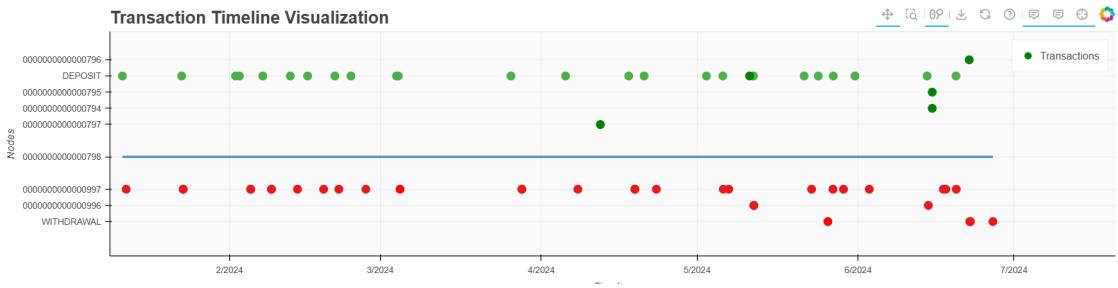


Figure 3.4: Timeline visualization for a money-mule target: the central blue line denotes the target entity, and each horizontal line represents a distinct counterpart. Incoming transactions are marked with green dots; outgoing transactions are marked with red dots. The interface is fully interactive, allowing users to pan and zoom to inspect different temporal segments.

3.6 TRACED Output/3: Detection of Suspicious Communities and Mule Risk Score

Aside from providing an innovative visualization framework for AML/CTF analysts, TRACED also performs data-driven, automated intelligence tasks to support analysts' work. In particular, it can identify and select suspicious transactions and counterparties associated with a given target, by leveraging graphs theory and machine-learning techniques. To this end, a new version of the algorithm proposed in [34] has been introduced —termed TRACED-FastMaN—enhancing and adapting the previous methodology for the specific AML/CTF scenario under consideration. The following sections highlight the key points in the original FastMaN methodology, and the difference and improvements proposed in TRACED-FastMaN.

3.6.1 FastMaN

FastMaN implements a topology-agnostic, temporal-graph approach to detect money-laundering flows across large transaction networks, as originally proposed

by Tariq and Hassani [34]. The algorithm begins by constructing a temporal graph in which each bank account is represented by a separate node at each distinct timestamp (or time bucket). In this representation, a directed edge from node A_t to node $B_{t'}$ exists if and only if account A transfers funds to account B at time $t < t'$.

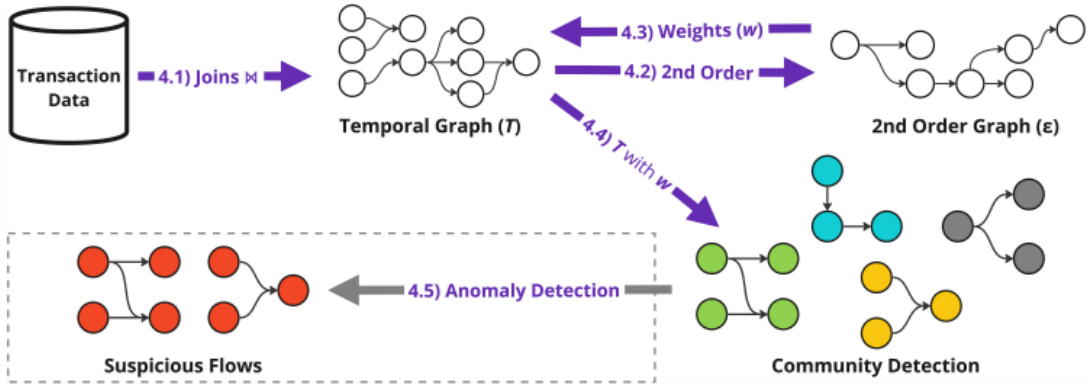


Figure 3.5: Original end-to-end architecture diagram of the FaSTMaN framework. The dashed grey section is not an integral part of the framework, and it is not taken into consideration in this work

To capture short-term sequential dependencies, FastMaN then builds a second-order transition graph: for each triple of consecutive transfers $A \to B$ at time t followed by $B \to C$ at time t', a directed edge is created between the ordered pair $(A \to B)$ and the next-hop pair $(B \to C)$. The weight of each second-order edge is defined as

$$W(A \to B,\, B \to C) \;=\; \max\Big(P(A \to B,\, B \to C),\, P'(A \to B,\, B \to C)\Big), \quad (3.1)$$

where

$$P(A \to B,\, B \to C) = \frac{\Big|S\Big(A \to B \;\sim\; B \to C\Big)\Big|}{\Big|S\Big(A \to B \;\sim\; B \to [*]\Big)\Big|} \quad (3.2)$$

$$P'(A \to B,\, B \to C) = \frac{\Big|S\Big(A \to B \;\sim\; B \to C\Big)\Big|}{\Big|S\Big([*] \to B \;\sim\; B \to C\Big)\Big|}. \quad (3.3)$$

Here,

- $S(A \to B \sim B \to C)$ denotes the set of all observed two-hop sequences $\big(A \to B,\ B \to C\big)$ in the dataset;
- $S(A \to B \sim B \to [*])$ denotes the set of all observed continuations of $A \to B$ to any subsequent transfer out of B;
- $S([*] \to B \sim B \to C)$ denotes the set of all observed two-hop sequences ending at $(B \to C)$ from any predecessor into B.

More generally, if we denote an arbitrary source-edge by $e_s = (f_s \to b_s)$ and a destination-edge by $e_d = (f_d \to b_d)$, the corresponding second-order weight is

$$W(e_s,\, e_d) \;=\; \max\Big(P(f_s \to b_s,\, f_d \to b_d),\, P'(f_s \to b_s,\, f_d \to b_d)\Big), \qquad (3.4)$$

using the analogous definitions of P and P' over sets of two-hop edge-pairs $\big(f_s \to b_s,\, f_d \to b_d\big)$.

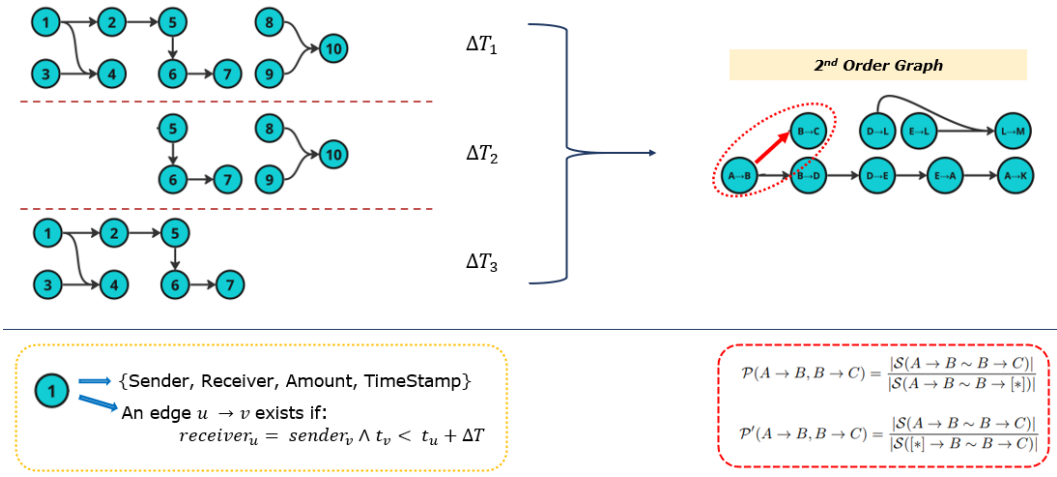


Figure 3.6: 2nd Order Graph construction workflow.

Transitions whose weights W fall below an adaptive threshold—determined by examining the empirical distribution of second-order counts—are pruned to eliminate spurious or one-off transaction patterns.

Once pruning is complete, FastMaN applies the Leiden community-detection algorithm to the weighted, pruned temporal graph. Each resulting community corresponds to a candidate laundering network, comprising accounts and time-ordered transfers that exhibit unusually dense connectivity. Because Leiden optimizes modularity on weighted graphs without requiring a predefined number of clusters or hop lengths, FastMaN remains agnostic to specific laundering typologies (e.g., number of interim "layering" accounts or "sink" endpoints).

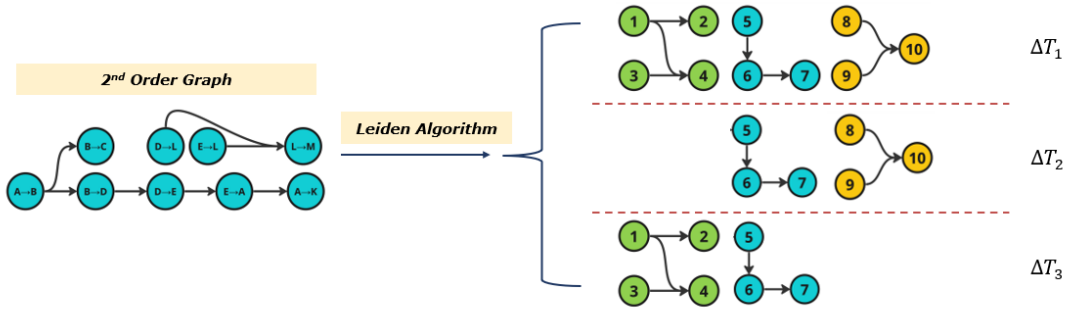


Figure 3.7: Weights computed on the 2nd Order Graph are backpropagated into the original graph and community detection is performed.

In the original evaluation, FastMaN was tested on a real-world dataset containing approximately 1.1 billion transactions aggregated from five major Dutch banks over a two-year period (January 2021–December 2022). FastMaN achieved over 95 % recall and 88 % precision at the 0.05 % injection level; by comparison, baseline approaches (e.g., FlowScope and motif-based database joins) attained only 60 %–70 % recall with significantly lower precision under identical conditions. On unlabeled, real-world data, FastMaN flagged approximately 2,400 high-risk communities, of which 390 (16.3 %) were subsequently confirmed by domain experts as genuine money-laundering flows. These results demonstrate that FastMaN's temporal, second-order methodology can scale to billion-edge graphs while maintaining high detection accuracy and low false-positive rates.

3.6.2 Limitations of FastMaN for TRACED's Egonet-Scale Analysis

The FastMaN architecture provides a robust starting point for the suspicious-transactions community-detection task outlined at the beginning of this section. In particular, it offers a feature-less, topology-agnostic metric that groups transactions based on their mutual co-occurrence in time. However, the original algorithm's context differs substantially from TRACED's. FastMaN is designed to analyze graphs on the order of billions of nodes and edges, leveraging a rich, complex topology to yield meaningful results even without using node or edge features. In TRACED's case, the algorithm must instead operate on the egonet of a selected target node (i.e., the node itself and all its direct counterparts). Such egonets typically contain only dozens to a few hundred nodes and edges—far less structure than FastMaN assumes—often resulting in uniformly assigned weights that hinder the quality of community assignments.

Furthermore, because FastMaN relies solely on topological structure and ignores

transaction features, its co-occurrence weights can be sometimes overly naive. For example, two transactions of €10 and €10 million could receive a weight of 1 if their timestamps align closely, causing them to fall into the same community. Such amount-agnostic grouping is unacceptable for money-mule detection, since a significant mismatch in transaction values generally indicates little to no direct correlation.

Finally, once communities are detected, the original algorithm does not provide a built-in method for distinguishing anomalous communities from legitimate ones, instead relying on more generic anomaly-detection strategies from the related literature.

3.6.3 TRACED-FastMaN

To address the issues reported above, several modifications are proposed in order to adapt FastMaN's weighting and community-detection mechanisms to TRACED's egonet-scale context, resulting in a new algorithm called TRACED-FastMaN.

- **Amount-based weights assignment:** the first improvement to the original algorithm is a direct modification of the weight assignment formulas reported in (3.2) and (3.3). Let

$$R_{\text{amt}}(A \to B,\, B \to C) \;=\; \frac{\min\Big(\text{amt}(A \to B),\, \text{amt}(B \to C)\Big)}{\max\Big(\text{amt}(A \to B),\, \text{amt}(B \to C)\Big)}.$$

  That is the amounts ratio between two possibly linked transactions. Then define

$$P^{\text{amt}}(A \to B,\, B \to C) = P(A \to B,\, B \to C) \;\times\; R_{\text{amt}}(A \to B,\, B \to C), \tag{3.5}$$

$$P'^{\text{amt}}(A \to B,\, B \to C) = P'(A \to B,\, B \to C) \;\times\; R_{\text{amt}}(A \to B,\, B \to C). \tag{3.6}$$

  Consequently, the adjusted second-order weight becomes

$$\begin{aligned} W^{\text{amt}}(A \to B,\, B \to C) = \max\Big\{ & P^{\text{amt}}(A \to B,\, B \to C), \\ & P'^{\text{amt}}(A \to B,\, B \to C)\Big\}. \end{aligned} \tag{3.7}$$

  In situations where the original weight would have been 1, the adjusted weight reaches at most $R_{\text{amt}}(A \to B,\, B \to C)$, that is the ratio of the two transactions' raw amounts. This guarantees that the algorithm never treats two transactions as more similar than their raw-amount ratio allows, resulting in more representative and robust weights across the network.

- **Linear regression–based score for anomaly scoring:** To separate legitimate targets from suspicious ones, a score comprising three elements is defined. These elements are computed at the target level, starting from the set of transactions that have been grouped together by the Leiden algorithm in the first-order graph. Let:

  n_t = number of communities found for target t,
  ρ_t = average in/out ratio across all communities of target t,
  a_t = total amount exchanged in all transactions flagged for target t.

  A higher n_t indicates a more suspect target, a lower ρ_t suggests a money-mule–like pattern, and a higher a_t implies greater suspicion. The final score is computed as a weighted sum of these three components:

  $$\text{Score}(t) \;=\; \beta_0 \;+\; \beta_1\, n_t \;+\; \beta_2\, \rho_t \;+\; \beta_3\, a_t\,. \tag{3.8}$$

  The coefficients $\beta_0, \beta_1, \beta_2, \beta_3$ are estimated by training a linear regression model on a dataset of legitimate and criminal targets drawn from the bank fraud case repository. Further details about the fitting process are provided in the Experiments section of this chapter.

3.7 Experiments

Experiments for this chapter are divided into two sections: the first estimates the model hyperparameters with the use of a huge synthetic transactional dataset to determine a suitable range of values, while the second leverages a dataset of legitimate and criminal targets—constructed with the aid of Intesa Sanpaolo fraud case repository—to estimate and test the weights of the linear regression task used for the final target-level score.

3.7.1 Hyperparameter Estimation

The hyperparameters required by TRACED-FastMaN are the same as those in the original algorithm:

- **Time Window δT:** the number of days over which to perform first-order graph aggregation.
- **In/Out Ratio of Communities ρ_{th}:** the ratio between total incoming and total outgoing transaction amounts for each community; any community exceeding the selected ratio is discarded by the algorithm when computing the anomaly score.

- **Weight Threshold W_{th}:** threshold applied to first-order graph edges after weight assignment; any edge with weight less than the selected value is discarded before community creation.

Dataset: The hyperparameter estimation in this work leverages the AMLworld synthetic dataset generator as described in Altman et al. [47]. This dataset comprises on the order of hundreds of millions of transaction records generated by an agent-based simulator calibrated to mirror real-world banking patterns (multiple banks, multiple currencies, various payment types). Each synthetic transaction includes a timestamp, source and destination account identifiers, transaction amount, currency, and payment method. Ground-truth labels indicate whether each transaction is part of a money-laundering pattern (e.g., fan-out, gather-scatter, cycle, bipartite, etc.), enabling precise evaluation of community-detection performance.

The choice to rely on a synthetic dataset rather than real ISP data arises from the need for transaction-level labels, rather than target-level labels. No existing in-bank dataset provides the granularity required to flag individual money-muling transactions; cases are typically reported only at the target or group level. Training hyperparameters at that higher level of aggregation would be unsuitable for this model's objectives. It is enough to consider that, even if a target is labeled as part of a money laundering scheme, not all transactions performed by him/her should be flagged as fraudulent: on the contrary, most of them will still be perfectly legitimate and only a small fraction will be part of the actual criminal operation due to the very nature of money-laundering operations (See Chapter 1). For this reason, training a model on such premises would inevitably yield an incorrect embedding. However, using synthetic data introduces its own challenges, such as distributional shifts between the synthetic training set and the real-world test set (since final predictions occur at the target level) and the generally lower fidelity of synthetic transactions—even in state-of-the-art generators like AMLworld [47]. The implications of these choices, as well as potential model enhancements and directions for future research, are discussed more in detail in the next section.

For each parameter, an array of candidate values is defined leveraging findings in [48] and domain-specific knowledge. The value ranges are reported in Table 5.1.

Table 3.1: Hyperparameter search ranges

Parameter	**Candidate Values**
`in_out_th` (In/Out ratio)	{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0}
`delta_t` (Time window δT)	{1, 2, 3, 4, 5, 6, 7, 8}
`cut_weight_th` (W_{th})	{0.00, 0.025, 0.05, 0.075, 0.01}

A GridSearch optimization is then performed over all combinations, and the

resulting F1 score is recorded for each. The results can be seen in 3.8, 3.9, 3.10 for the three best values of cut-weight threshold. Other values yield almost random classification and are not displayed.

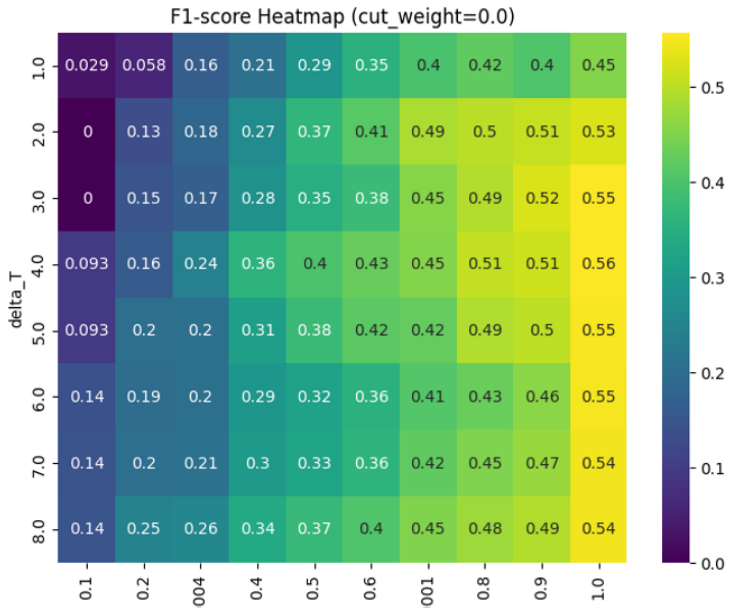


Figure 3.8: F1 score for δT and ρ_{th} at cut-weight 0.0

Results indicate that the algorithm struggles to robustly identify fraudulent transactions when using the smallest time window (δT) combined with a very strict in/out amount ratio. This behavior is expected given the nature of the egonet graphs. As noted above, target egonets are typically small, and partitioning them into too many time buckets produces first-order graphs that are too sparse to capture meaningful temporal patterns. Likewise, enforcing a very tight in/out ratio excludes many legitimate laundering structures and reduces coverage of diverse ML schemes. However, when these constraints are relaxed slightly, the algorithm demonstrates a strong ability to distinguish fraudulent from legitimate transactions, achieving an F1 score near 0.7 across multiple hyperparameter combinations. This finding suggests that TRACED, with appropriately tuned settings, can serve as a reliable tool for isolating communities of related transactions.

3.7.2 Linear Regression and Target Accuracy

To assess TRACED's capability of separating money-mule targets from legitimate ones, a linear regression model was trained on a dataset of real transactions taken from 25 fraud cases identified by Italian law enforcement agencies and stored by

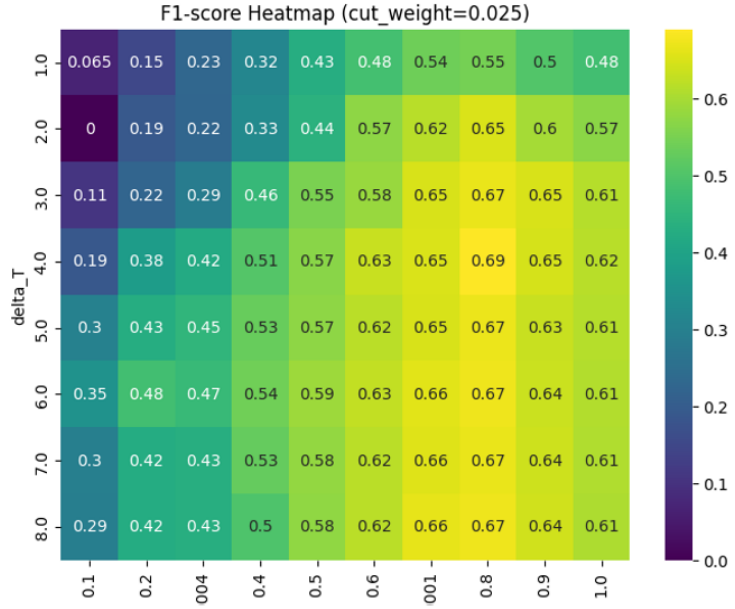


Figure 3.9: F1 score for δT and ρ_{th} at cut-weight 0.025

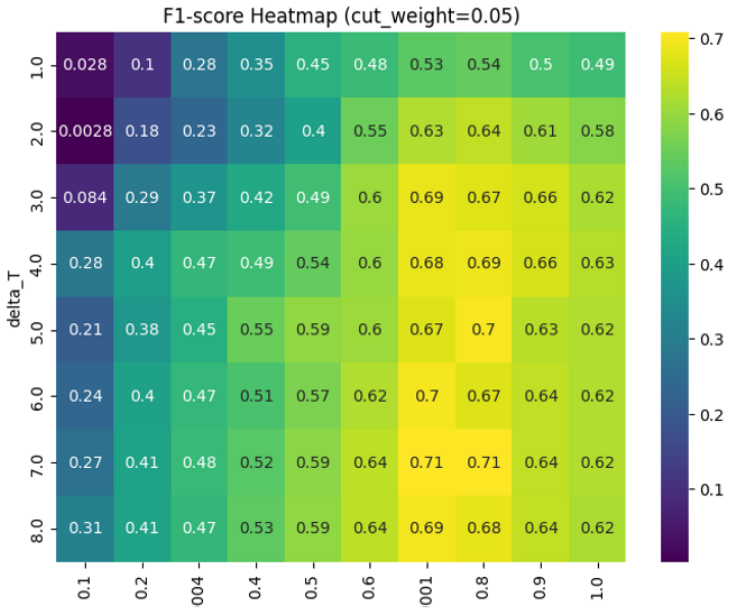


Figure 3.10: F1 score for δT and ρ_{th} at cut-weight 0.05

the bank between March 2023 and January 2025. These cases were injected into a dataset of 1,000 legitimate targets drawn from normal bank operations. The

combined dataset was split 70/30 into train and test sets, and the linear regression model was trained and evaluated using the ROC AUC metric. The resulting weights and intercept are shown in Table 3.2.

Table 3.2: Linear regression weights and intercept

Parameter	**Value**
β_1 (number of communities)	1.48845
β_2 (average in/out ratio)	–0.66195
β_3 (total amount)	–0.16766
β_0 (intercept)	–0.62699
ROC	0.856

Training the linear regression score on real-world labeled fraud and non-fraud cases guarantees that each learned coefficient β_n accurately captures the predictive importance of its associated metric—whether community count, aggregate balance, or total exchanged amount—so that the resulting risk score not only discriminates effectively between legitimate and illicit profiles, reaching an AUC-ROC of almost 0.86, but also remains interpretable and robust to overfitting through appropriate regularization.

Leveraging these learned coefficients, TRACED assigns to each target a risk score computed from the three selected metrics, each one weighted by its corresponding learned β_n parameter. This real-valued, ML-based score integrates with TRACED's visualization tools, providing analysts with both intuitive visual narratives and data-driven risk assessments to inform their investigative decisions. In its raw form, this score facilitates direct comparison between targets (with higher values indicating greater suspicion). Alternatively, it can be normalized to a fixed interval—for example by applying a softmax to map scores into [0,1]—to produce an absolute suspicion grade for any single target.

3.8 Conclusions on TRACED

Summing up the procedures and strategies listed in the previous sections, TRACED proposes itself as a lightweight, egonet-scale community-detection framework that combines temporal-graph topology with transaction-level features to isolate money-mule patterns in banking data. By adapting FastMaN's second-order weighting to include amount-based adjustments, TRACED is able to operate on small egonets (tens to hundreds of nodes) while still capturing meaningful temporal and value-based correlations. Its clustering algorithm approach ensures robust community identification, and the subsequent linear-regression scoring model prioritizes truly

suspicious targets based on the number of communities, in/out-ratio patterns, and total exchanged amounts. Altogether, TRACED offers a practical, tunable solution for banks to detect money-muling activity at the egonet level, achieving strong separation between legitimate and fraudulent targets even when only limited local structure is available. Moreover, TRACED's interactive visualization tools—anchored around a zoomable, timeline-centric representation of transactional flows and enriched with relational community overlays—greatly enhance result interpretability: investigators can intuitively trace fund movements, filter and drill into high-risk flows, and export annotated timelines and pattern summaries directly into intelligence reports. Supported by its promising results on synthetic data and past fraud cases, TRACED is currently operational in the TXM pipeline of Intesa Sanpaolo: the months following the publication of this work will serve to test its real effectiveness on supporting AML/CFT analysts on the real field of TXM.

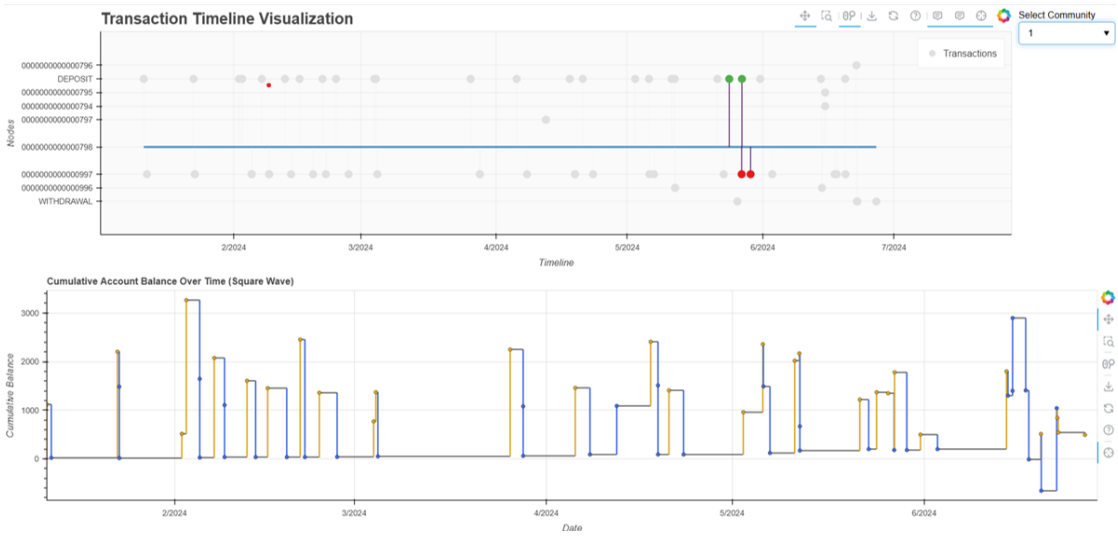


Figure 3.11: Overview of the user interface: communities can be highlighted separately to analyze different temporal sections of the plot

Chapter 4

Deep Learning on Temporal Graphs and Sampling Algorithms

This chapter provides an overview of the fundamental concepts behind the second model proposed in this thesis, beginning with a review of Artificial Neural Networks (ANN) principles and Deep Learning (DL) methodologies. It then offers a formal definition and basic notations of temporal graphs—networks whose nodes, edges, and attributes evolve over discrete time steps—and surveys specialized deep architectures for capturing their dynamics, including Temporal Graph Neural Networks and recurrent graph models. Temporal Networks represent a robust and very well suited structure to model the problem under investigation, capturing the dynamic and evolving nature of financial transactions with greater accuracy [21]. Next, graph downsampling algorithms are introduced as means of reducing the size and complexity of large transactional graphs data used for models' input. Leveraging those algorithms and applying transfer-learning principles, these techniques enable scalable, efficient training for financial transaction forecasting and anomaly detection in large-scale financial flows.

4.1 Introduction to ANN

The field of artificial neural networks (ANNs) traces its origins to the perceptron (see Fig 4.1), introduced by Frank Rosenblatt [49] in 1958 as "the first machine which is capable of having an original idea" – a single-layer binary classifier that learned to distinguish left-marked from right-marked signs on punch cards.

Despite its promise, the perceptron's inability to solve non-linearly separable

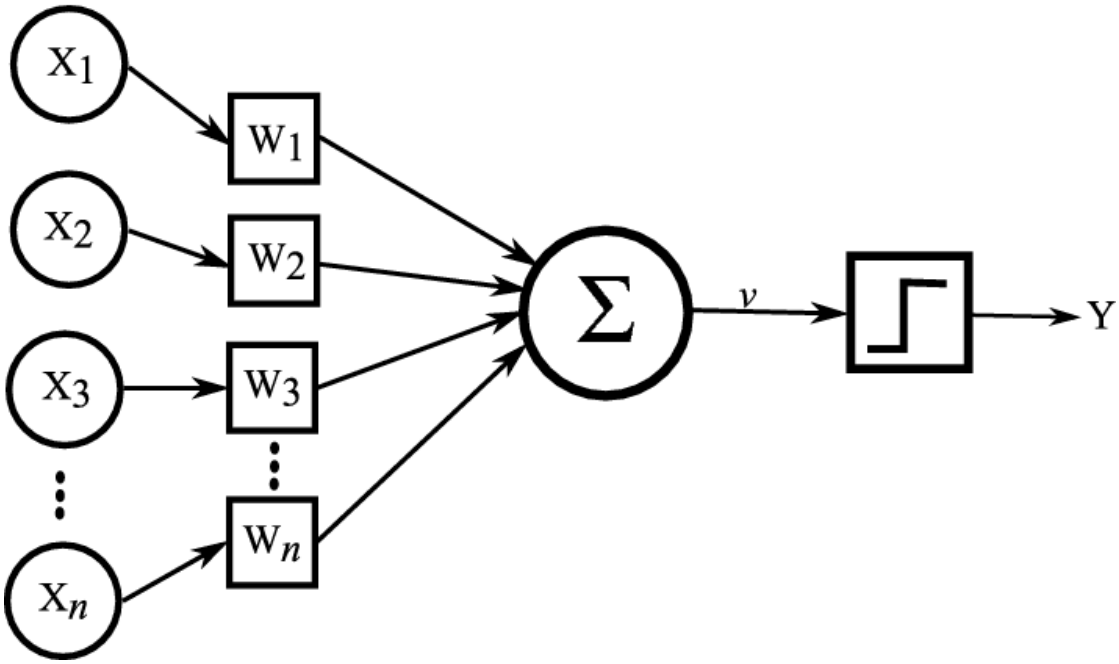


Figure 4.1: The original perceptron architecture proposed by Rosenblatt in 1958: weights are learned and then passed into a non-linearity activation function to approximate the desired function.

problems led to skepticism and a temporary decline in ANN research. This impasse was broken nearly three decades later by the invention of the backpropagation algorithm, which enabled efficient training of multi-layer perceptrons (MLPs) via gradient descent on a differentiable loss function. In 1986 Rumelhart, Hinton, and Williams demonstrated that backpropagation could learn internal representations in deeper networks, publishing the landmark paper "Learning representations by back-propagating errors"[50] .

However, simply adding layers did not immediately yield better models: training very deep networks remained challenging due to vanishing gradients and local minima. A major turning point came in 2012, when AlexNet [51] was proposed: a deep convolutional neural network for image classification trained on the ImageNet dataset that reduced top-5 error rates from 26.2% to 15.3%, demonstrating the power of large-scale, GPU-accelerated deep models for complex pattern recognition.

Since then, neural networks have experienced an extraordinary improvement, driven by advances in algorithms, hardware, and data availability. Researchers scaled up deep architectures—convolutional nets for image understanding, recurrent and transformer models for language, and graph neural networks for relational data—fueled by the parallel compute power of GPUs and, more recently, TPUs. This revolution has touched virtually every domain, from computer vision to natural

language processing to generative tasks such as image synthesis, text and code generation with transformer-based language models, music and speech synthesis, video generation, and even molecular graph design.

4.2 ANN Structures and Learning

Modern ANNs models rely on a handful of canonical building blocks and a uniform learning framework based on gradient-based optimization. The following paragraphs provide a brief functional overview of the most widely used modules, with particular emphasis on those central to the following sections of this work.

4.2.1 Convolutional Blocks

One of the first and most used architectural blocks employed in ANNs are convolutional block: they apply a set of learnable filters (kernels) to their input feature maps, producing new feature maps that encode localized patterns. Formally, given an input tensor $\mathbf{X} \in \mathbb{R}^{H \times W \times C_{\text{in}}}$ and a bank of C_{out} kernels $\{\mathbf{K}^k\} \subset \mathbb{R}^{h \times w \times C_{\text{in}}}$, each output channel is

$$Y^k_{i,j} \;=\; \sum_{c=1}^{C_{\text{in}}} \sum_{u=1}^{h} \sum_{v=1}^{w} K^k_{u,v,c}\, X_{\,i+u-1,\; j+v-1,\; c} \;+\; b^k \,,$$

where b^k is a bias term. A non-linear activation (e.g. ReLU: $\max(0, \cdot)$) follows each convolution to introduce model capacity on the feature maps beyond linear filters. Pooling layers (max or average) optionally reduce spatial resolution by aggregating over local neighborhoods. By stacking multiple convolutional blocks, deeper networks learn hierarchical representations, from edges in early layers to high-level semantics in later layers (see Fig 4.2).

4.2.2 Recurrent Blocks

Recurrent blocks are used to create Recurrent Neural Networks. They define a new variable, called hidden state $\mathbf{h}_t \in \mathbb{R}^H$, i.e. a vector encoding the network's internal memory at time t, updates as

$$\mathbf{h}_t = \phi\Big(W_{hh}\,\mathbf{h}_{t-1} + W_{xh}\,\mathbf{x}_t + \mathbf{b}_h\Big), \quad \hat{\mathbf{y}}_t = W_{hy}\,\mathbf{h}_t + \mathbf{b}_y,$$

where W_{hh} propagates information from the previous state, W_{xh} projects the current input $\mathbf{x}_t$, $\mathbf{b}_h$ and $\mathbf{b}_y$ are bias vectors, and ϕ is an element-wise nonlinearity; by iterating this compact update the model continuously blends past context and new observations into $\mathbf{h}_t$, enabling it to embed temporal patterns at multiple levels of

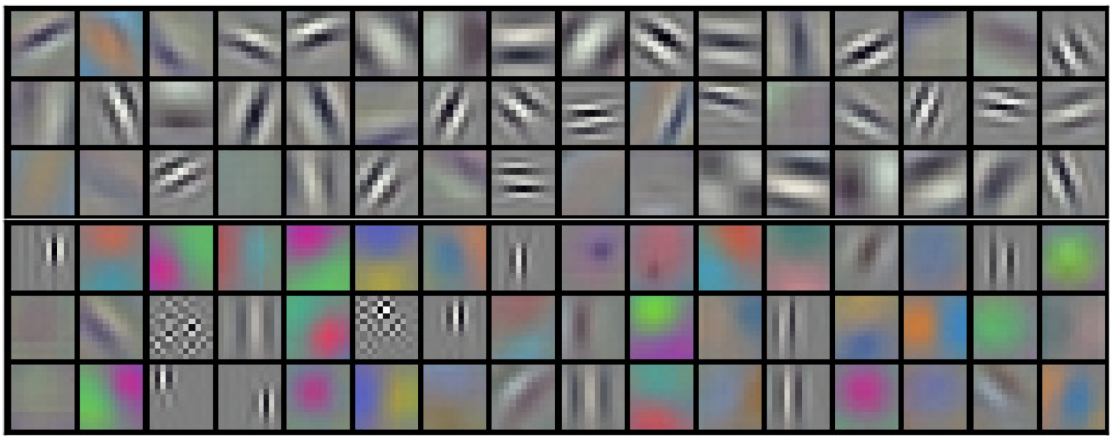


Figure 4.2: The 96 "features detectors" learned by AlexNet [51]: the network can tell for each point of an input image "how much" it resembles each one of the filters, and hierarchically build complex features for classification.

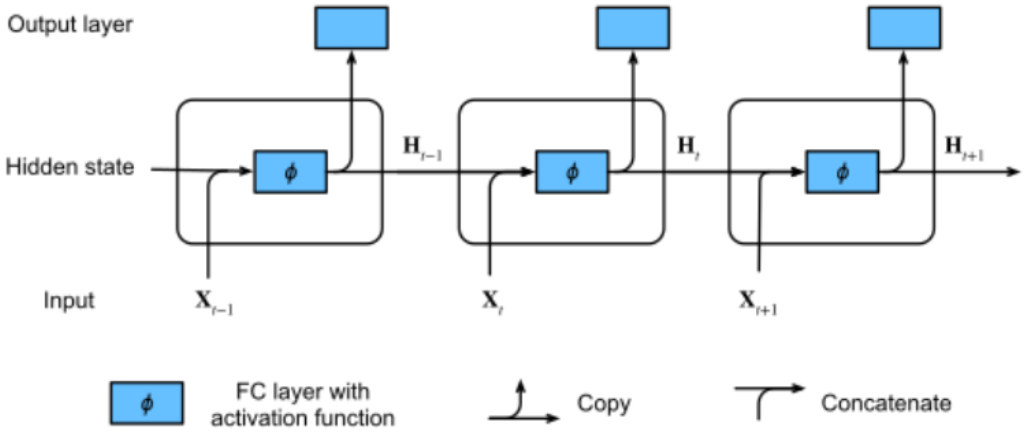


Figure 4.3: RNN Unit architecture.

abstraction before decoding them into $\hat{\mathbf{y}}_t$. The typical architecture of an RNN cell is shown in Fig 4.3

In recurrent neural networks (RNNs), problems related to the training process can be more severe than in convolutional architectures; to address this, gated variants such as the Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) introduce multiplicative gating mechanisms that dynamically regulate information flow and gradient propagation across time steps.

LSTM Long Short-Term Memory units extend vanilla RNNs with gating mechanisms that control the flow of information and enable the network to learn long-range

dependencies. At each time step, an input gate decides how much new information enters the memory cell, a forget gate determines which past information to discard, and an output gate controls what is passed to the next layer. This structure allows LSTMs to capture both short-term fluctuations and longer-term patterns in sequential data without suffering from vanishing gradients.

GRU Gated Recurrent Units simplify the LSTM architecture by merging the input and forget gates into a single update gate, and by using a reset gate to modulate how much of the previous hidden state contributes to the candidate activation. With fewer parameters and a more streamlined design, GRUs often match LSTM performance while training faster and requiring less memory.

4.2.3 Activation Functions and Pooling Layers

Non-linear activations are usually applied immediately after each convolution to introduce non-linear model capacity. Common choices include the Rectified Linear Unit (ReLU: $\max(0, x)$), which accelerates convergence and mitigates vanishing gradients, and its variants (leaky-ReLU, ELU). Sigmoid and hyperbolic-tangent functions were historically popular but are now less common in deep architectures due to saturation effects.

Pooling layers optionally follow activations to reduce spatial dimensions and introduce translation invariance. Max pooling selects the largest activation over non-overlapping windows, preserving the strongest response, while average pooling computes the mean. Pooling both reduces computational cost and helps the network build hierarchical, multi-scale representations.

4.2.4 Learning Framework and PAC Guarantees

Training an artificial neural network involves finding parameters $\theta \in \mathbb{R}^P$ that minimize the empirical risk

$$L(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell\Big(y_i, \hat{y}(\mathbf{x}_i; \theta)\Big),$$

where n is the number of training examples, $\ell(y, \hat{y})$ is a pointwise loss on the network outputs (e.g. cross-entropy $\ell(y, \hat{y}) = -\sum_k y^{(k)} \log \hat{y}^{(k)}$ or mean-squared error $\ell(y, \hat{y}) = \|\hat{y} - y\|^2$), and $\hat{y}(\mathbf{x}_i; \theta)$ denotes the network's prediction for input $\mathbf{x}_i$. Optimization proceeds by stochastic gradient descent, which at each iteration computes the gradient of L with respect to θ via backpropagation and updates

$$\theta \;\leftarrow\; \theta - \eta \, \nabla_\theta L(\theta),$$

where $\eta > 0$ is the learning rate controlling the step size; this update moves θ in the direction of steepest descent on the loss surface, allowing the model to gradually adjust its weights to better fit the data over successive passes.

The dataset is commonly split into three parts:

- **Training set:** used to update the model parameters via (stochastic) gradient descent.
- **Validation set:** used to monitor performance during training, tune hyperparameters, and decide when to stop (early stopping).
- **Test set:** held out until final evaluation to provide an unbiased estimate of how the model will perform on new data.

While deep learning's success is largely empirical, statistical learning theory provides *Probably Approximately Correct* (PAC) guarantees. Informally, if a model class has limited complexity relative to the number of training examples, then—with high probability—a model that achieves low error on the training set will also have low error on unseen data. In the Probably Approximately Correct (PAC) framework, if the hypothesis class $\mathcal{H}$ has VC–dimension d_{VC}, then with probability at least $1 - \delta$ over n i.i.d. samples, true risk is $R(\theta) \leq \epsilon$ with confidence $1 - \delta$.

To reach this guarantee, it suffices to collect

$$n \;=\; O\Big(\tfrac{1}{\epsilon^2}\Big[d_{VC}\log\tfrac{1}{\epsilon} + \log\tfrac{1}{\delta}\Big]\Big)$$

training examples and find θ with $\hat{R}_n(\theta) \leq \epsilon/2$

These bounds depend on measures of hypothesis-class complexity (such as VC-dimension or Rademacher complexity) and quantify how much data is needed to generalize to a desired accuracy with a specified confidence, while giving mathematical guarantees about the optimality and feasibility of ANN based solutions.

4.3 ANNs models for Temporal Graphs

In this work, Temporal Graphs can be employed to model the evolving set of financial players and interactions between them in the form of transactions managed by the bank. This data can be stored in a graph structure, leveraging similar concepts and tools as the one listed in Chapter 3.

To detect and predict anomalies in such temporal graphs, graph-based recurrent neural networks provide an effective solution. These models employ specialized encoders that extend convolutional and recurrent building blocks to irregular graph domains by replacing standard matrix–vector products with neighborhood aggregation via graph convolutions. Such encoders learn node representations that

respect the evolving topology, and when coupled with temporal units (e.g. GRUs or LSTMs) they jointly capture spatial dependencies in the graph and temporal patterns in node and edge features.

4.3.1 Temporal Graphs

Temporal graphs (also known as dynamic or time-varying graphs) extend static graphs by capturing how relationships appear and disappear over discrete time steps [52]. Formally, let $G = (V, E)$ be a static graph. A *temporal graph* is given by a labeling

$$\lambda \colon E \to 2^{\mathbb{N}},$$

that assigns to each edge $e \in E$ the set of time-labels at which e is active. Equivalently, one can view a temporal graph as a sequence of static snapshots

$$D(t) = \Big(V,\, A(t)\Big), \qquad A(t) = \{\, e \in E : t \in \lambda(e)\},$$

facilitating analysis with classical graph-theoretic tools. In this thesis, temporal graphs are modeled leveraging the sequence of static snapshots approach (see Fig. 4.4).

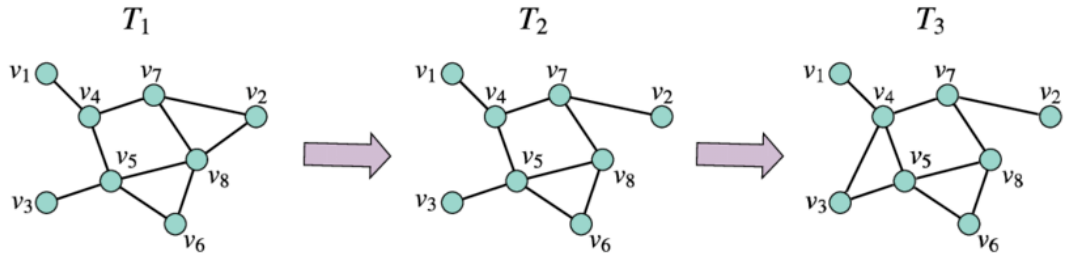


Figure 4.4: Example of evolving set of nodes/edges in different timestamps of a Temporal Graph

4.3.2 R-GNN Encoders Architectures

Several graph-based recurrent architectures (R-GNN) have been proposed for spatio-temporal modeling on dynamic graphs. Most models rely on a shared spatial operator—*graph convolution*, typically defined as:

$$G(X, A) = \sigma\left(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}XW\right)$$

where $\hat{A} = A + I$ adds self-loops to the adjacency matrix, $\hat{D}$ is the corresponding degree matrix, X is the input feature matrix, W is a learnable weight matrix, and

$\sigma(\cdot)$ denotes a non-linear activation function. By normalizing and aggregating neighbor features in this way, the convolutional stage produces embeddings that reflect each node's local topology. These embeddings are then fed into recurrent components—such as gated recurrent units (GRUs), long short-term memory (LSTM) cells, or graph-specific recurrent cells—that learn the temporal evolution of features across successive graph snapshots.

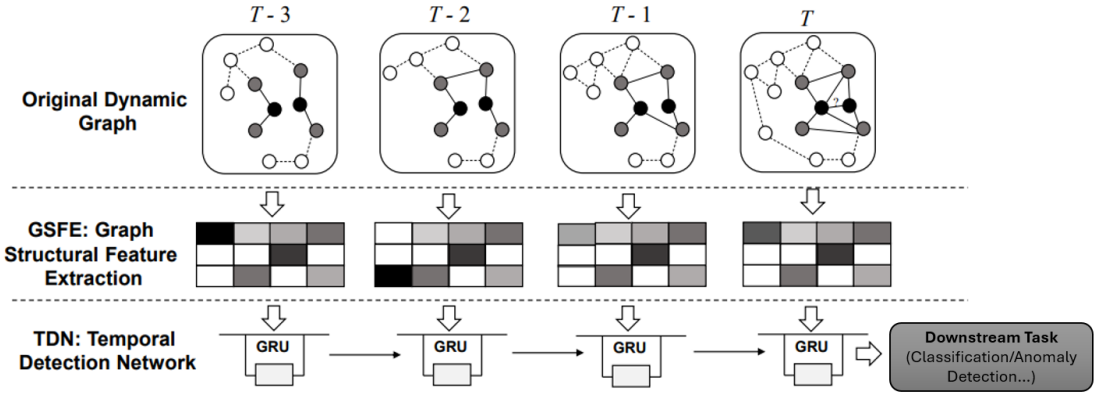


Figure 4.5: Example of operative pipeline of a G-RNN model: the first step learns structural dependencies and the second combines them to learn temporal patterns.

4.3.3 Related Works

Several G-RNN encoders have been proposed in literature. **GConv–GRU** integrates the graph convolutional layer into the update equations of a Gated Recurrent Unit, allowing hidden states to propagate across edges before gating; by embedding neighbor information directly into both the reset and update gates, this architecture enhances spatial–temporal representation learning and has been validated on traffic speed forecasting in urban road networks, yielding significant reductions in prediction error compared to vanilla GRUs [53]. **GConv–LSTM** applies graph convolutions in place of the linear transformations within each LSTM gate and the candidate cell update, thereby preserving spatial locality and enabling longer-range dependency modeling; its efficacy has been demonstrated on skeleton-based human action recognition and short-term traffic prediction tasks [53]. **GCLSTM** interleaves multiple graph convolutional layers around the input, forget, and output gates—stacking spatial filters both before gate activations and after the cell update—to capture higher-order neighborhood structures within each time step; this deep spatio-temporal filtering mechanism has shown marked improvements in anomaly detection for power-grid stability and traffic congestion modeling [54]. **LRGCN** augments standard GCNs with multi-hop neighborhood aggregation

and residual connections across successive time slices, effectively capturing both local and global graph patterns in a single forward pass; applied to traffic forecasting and water distribution network analysis, it mitigates over-smoothing and supports deeper spatial learning without degradation [55]. **DyGR** alternates between graph convolutional updates and parameter evolution via recurrent networks, adapting convolutional kernels to the evolving topology and preserving temporal context; evaluated on dynamic citation and social-interaction networks, this approach outperforms static GCNs in link prediction and node classification under changing conditions [56]. Finally, **TGCN** couples a graph convolutional layer with a causal 1D convolution over time steps—eschewing recurrent units in favor of parallelizable temporal convolutions—offering reduced training latency and robust performance; originally introduced for traffic speed forecasting in Beijing and Los Angeles road networks, it achieves competitive accuracy while enabling efficient, real-time deployment [57].

4.4 Graph Sampling Algorithms and Transfer Learning

When working on large-scale financial flows modeled as graphs structures, scale and complexity of data can become a huge challenge [58]. These graphs can easily span thousands of nodes and tens of millions of edges, pushing the limits of memory and computation on a single machine and often necessitating costly distributed or parallel infrastructures. Moreover, ANN-based solutions are typically data-hungry at the training stage, requiring extensive information to robustly model complex, long-range dependencies inherent in transaction sequences.

Transfer Learning and Graph Sampling Algorithms offer a promising way to mitigate these challenges: by training models on much smaller, representative samples of the original graphs and then transferring the learned parameters to full-scale data at test time, it is possible to preserve accuracy while dramatically reducing computational overhead. To introduce this intuition in practical TXM pipelines, this section presents an overview of several graph-sampling strategies drawn from the scientific literature.

4.4.1 Transfer Learning

Transfer learning leverages knowledge acquired from a source domain D_S and task T_S to improve model performance on a related target domain D_T and task T_T, especially when labeled data in the target is scarce or data size becomes a challenge [59]. Formally, a domain D is characterized by its feature space $\mathcal{X}$, and a task T by its label space $\mathcal{Y}$ and prediction function f. In practice, a model is

first pre-trained on a smaller (D_S, T_S), learning representations or parameters that capture general patterns, and then fine-tuned or directly transferred on the more challenging (D_T, T_T), thus reducing the need for extensive and heavy tranining frameworks and accelerating convergence to an accurate solution. .

4.4.2 Graph Sampling Methods

Graph sampling methods may pursue several objectives, such as obtaining a smaller yet representative set of vertices, preserving key structural characteristics of the original graph, or even generating new random instances based on a larger input. In this work, the primary goal is to minimize the divergence between the distributions of selected graph properties in the sample and in the original. By doing so, it becomes possible to train a neural-network model on a substantially reduced graph, thereby achieving notable savings in training time and computational resources while retaining sufficient generalization to transfer learned parameters back onto the full-scale graph with only minor adjustments or fine-tuning.

In the related literature [60], sampling algorithms are typically classified into three families based on their selection mechanism:

- *Vertex-based sampling*: selects a subset of vertices and induces the subgraph on them.
- *Edge-based sampling*: selects a subset of edges and induces the subgraph on their endpoints.
- *Traversal-based sampling*: grows the sample by walking through the graph, following its connectivity.

The following paragraphs provide a brief overview of some algorithms belonging to the three categories cited above.

Uniform Vertex Sampling (VS) Selects k vertices uniformly at random from V and forms the sample by inducing the subgraph on those vertices. Formally, given $G = (V, E)$ with $|V| = n$ and $k < n$, choose $V_s \subset V$, $|V_s| = k$, uniformly; then $G_s = (V_s, E_s)$ with $E_s = \{(u, v) \in E : u, v \in V_s\}$ [60].

Vertex Sampling with Neighborhood (VSN) Begins by selecting k seed vertices uniformly, then adds their immediate neighbors—optionally up to r per seed—to form V_s, and induces E_s on V_s. This preserves more local connectivity at the cost of a larger sample [60].

Edge Sampling (ES) Chooses q edges uniformly at random from E and lets V_s be the set of their endpoints; the sampled graph is $G_s = (V_s, E_s)$ with E_s the selected edges [60].

Breadth-First Sampling (BFS) Starts from a random seed v_0 and explores neighbors layer by layer (FIFO queue) until k vertices are visited, then induces the subgraph. BFS preserves connectivity and short-path properties but tends to over-sample dense regions [61].

Depth-First Sampling (DFS) Uses a LIFO stack from a random seed v_0, diving along a path until no new neighbors remain, then backtracking, until k vertices are sampled and the subgraph induced [61].

Forest Fire Sampling (FFS) Simulates a "burning" process: each burned node selects a geometric(p) number of unburned neighbors to burn next, repeating until k vertices, then induces the subgraph. The burn parameter p controls exploration width, balancing local clustering and component-size fidelity [62].

Snowball Sampling (SNOW) Proceeds in waves from a seed v_0; each frontier node recruits up to r unvisited neighbors per wave until k vertices are collected, then induces the subgraph. Limiting recruits per node mitigates high-degree bias [63].

Random-First Sampling (RFS) Maintains a frontier like BFS but selects the next vertex to expand uniformly at random from the frontier. Continues until k vertices sampled, then induces the subgraph. RFS balances connectivity preservation with reduced layer bias [61].

Random Walk Escape (RWE) Alternates between local random-walk steps and global jumps: with probability α, jumps to a new random node; otherwise moves to a random neighbor. After k distinct vertices, induces the subgraph. The escape rate α tunes the mix of uniformity and locality.

In Chapter 5, these sampling algorithms will be evaluated as preprocessing strategies for the R-GNN anomaly detection model, measuring their ability to produce reduced yet representative subsets of the original graph data and quantifying the resulting gains in training scalability—namely, reductions in runtime and memory footprint—while ensuring sufficient fidelity for accurate model learning.

Chapter 5

GRAND-Net: Graph Recurrent Anomaly Detection in Large-Scale Financial Flows

The second major contribution of this thesis is the design and implementation of GRAND-Net, a novel recurrent graph neural network architecture designed to model data from bank's large-scale financial flows and perform anomaly detection on the generated embedding. GRAND-Net is trained to both forecast future transaction volumes at an high aggregation level, and also to flag abrupt, potentially anomalous shifts in those flows as they occur.

GRAND-Net addresses the need for obliged entities such as Intesa Sanpaolo to continuously oversee the integrity of all managed transactions, uncover recurring patterns across multiple levels of abstraction, and detect shifts in those patterns that may signal illicit fund movements—whether large-scale money-laundering networks or terrorism-financing schemes. Ideally, this monitoring should be robust, adaptable, and largely automated, sparing analysts and domain experts from the difficult and time-consuming task of manually sifting through vast data in search of suspicious entities or behaviors.

Starting from the collection of all transactions managed by the bank on a given period of time, GRAND-Net is able to map them in a graph structure, produce a robust internal embedding of the relational and temporal features of those transactions, and finally produce forecasts for upcoming intervals, predicting the evolving graph topology and the distribution of transaction volumes on each edge, and finally compute an anomaly score on edges based on learned features for

analysts support.

To produce those outputs, this work proposes leveraging Artificial Neural Networks (ANN), and specifically graph-based models, to build a tool capable of real-time anomaly detection on temporal transaction data. The next sections provides a detailed description of the proposed model's architecture and feature design, including how transaction flows are ingested, processed, and scored at multiple levels of abstraction. A comparative study then evaluates several encoder–decoder combinations on real bank data, assessing their ability to detect anomalous patterns and shifts in macroscopic financial flows.

5.1 GRAND-Net Design and Workflow

The proposed model is a ANN that is able to integrate spatiotemporal graph convolution with gated recurrent modules and a unified edge-scoring mechanism—combining edges existence probability and weight prediction—to directly identify anomalous edges in dynamic financial networks. The architecture of the model can be seen in 5.4. Extensive experiments, provided in the following sections, demonstrate that this approach significantly outperforms existing methods under varying anomaly-injection rates.

The proposed model comprises two principal components. The first is a graph-based recurrent neural network (R-GNN) that consumes a sequence of historical transaction graphs

$$G_{T-N},\, G_{T-N+1},\, \ldots,\, G_{T-1},$$

each encoding the topology of transactions processed by the bank over a desired time window, and is trained to reconstruct the future transaction graph $\hat{G}_T$ at time step T by combining outputs from two parallel heads: one for link prediction and the other for edge-weight regression.

In the second stage, the pretrained embeddings of each temporal snapshot are used to evaluate test graphs into which synthetic anomalies have been injected. A regression-based decoder then predicts edges existance probability and weights and compares them to the observed values, assigning each edge an anomaly score that quantifies its deviation from expected behavior—thereby enabling the identification of edges most likely involved in money-laundering or terrorist-financing schemes.

5.2 Data Loading and Preprocessing

GRAND-Net can ingest any transactional dataset that provides a minimal set of required fields—namely sender, receiver, transaction timestamp, and transaction amount. It then maps these transactions onto a graph structure (as described in Chapter 3), enabling aggregation of nodes and edges at multiple levels of abstraction.

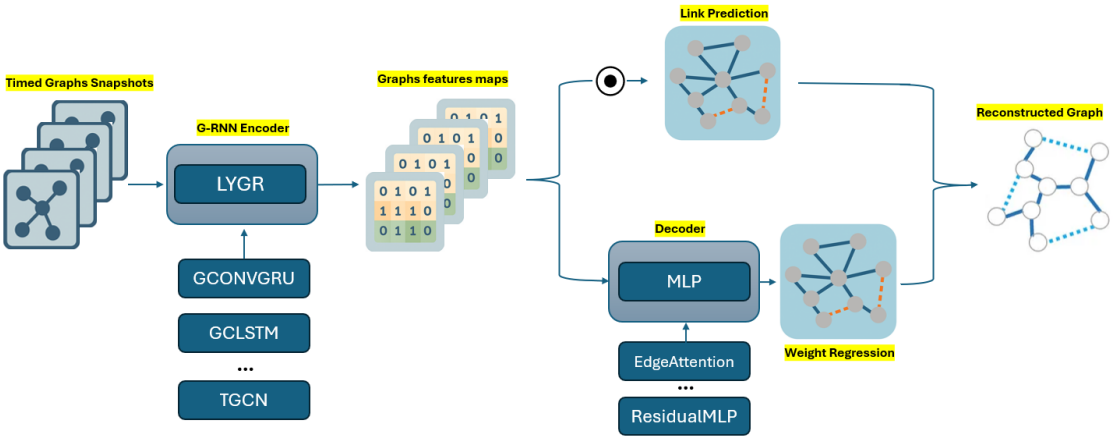


Figure 5.1: R-GNN proposed Encoder-Decoder architecture.

In this work, we use each transaction's sender and receiver BIC code as the node identifier. By selecting different character ranges within the BIC (see Fig. 5.2), we can flexibly adjust geographic granularity—for example, using characters 1–6 to group all ISP offices in Italy or characters 6–9 to isolate branches in Milan.

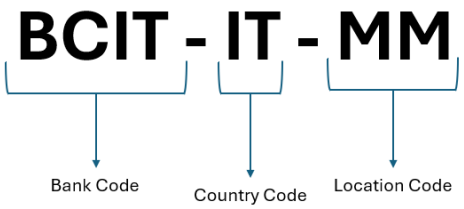


Figure 5.2: BIC code example for Intesa Sanpaolo's Italian branch in Milan. By selecting different character ranges within the code—e.g. characters 1–6 to group all ISP offices in Italy, or characters 6–9 to isolate all Milan branches—the model can perform analyses at varying levels of geographic granularity.

Once the desired geographic granularity is set, we partition the graph into successive temporal snapshots, yielding a dynamic representation of money flows. This evolving graph allows the model to learn both topological and temporal relationships among entities, producing embeddings that feed into downstream prediction tasks.

5.2.1 Anomalies Injection

Once the model has learned robust temporal and relational embeddings, it can be applied to detect anomalies in future transaction graphs. However, as noted in Chapter 2, Section "Related Works," publicly available datasets with labeled financial-transaction anomalies are scarce and Intesa Sanpaolo's internal records currently anomalies labeled at transaction level taken from real fraud cases. Consequently, this work relies on synthetic anomaly generation. Combining intuitions taken from both related literature and domain knowledge, three classes of anomalies are injected into the graph:

- **Structural anomalies:** these affect the topology of the graph. Following [64, 65], m random nodes are selected and fully connected to form a *clique*—a structure widely considered to be anomalous in many contexts, including financial fraud related ones. Repeating this n times yields $n \times m <$ injected cliques.
- **Contextual anomalies:** those are represented by edges whose weights deviate significantly from the model's prediction despite normal connectivity patterns (e.g. sudden and unusually large or small transaction amounts). They are obtained as in 5.4, by perturbing the edge-weight features: n edges are selected, and for each one its weight is substituted by the most distant one randomly chosen among k candidates in the network.
- **Domain-based anomalies:** following domain knowledge acquired from the ISP office, this anomaly models sanction-evasion routing. When economic exchange between two nations is restricted (e.g. due to embargos, wars, or tariffs), transactions are often funneled through third-party jurisdictions, an illegal practice that has led private companies and obliged entites to fines and investigations in the past[1]. To emulate this behavior, a pair of nodes is randomly selected and its exchanged amount in the real graph is perturbed by rerouting it along low-probability multi-hop paths in the network.

5.3 GRAND-Net Model Definition and Training Pipeline

Once the structures, inputs and outputs of the model have been declared, it is necessary to perform a series of task to define and optimize model's training

[1] `https://english.scenarieconomici.it/economy-and-business/volkswagens-shadow-play-german-jetta-brand-defies-sanctions-returns-to-russia/`

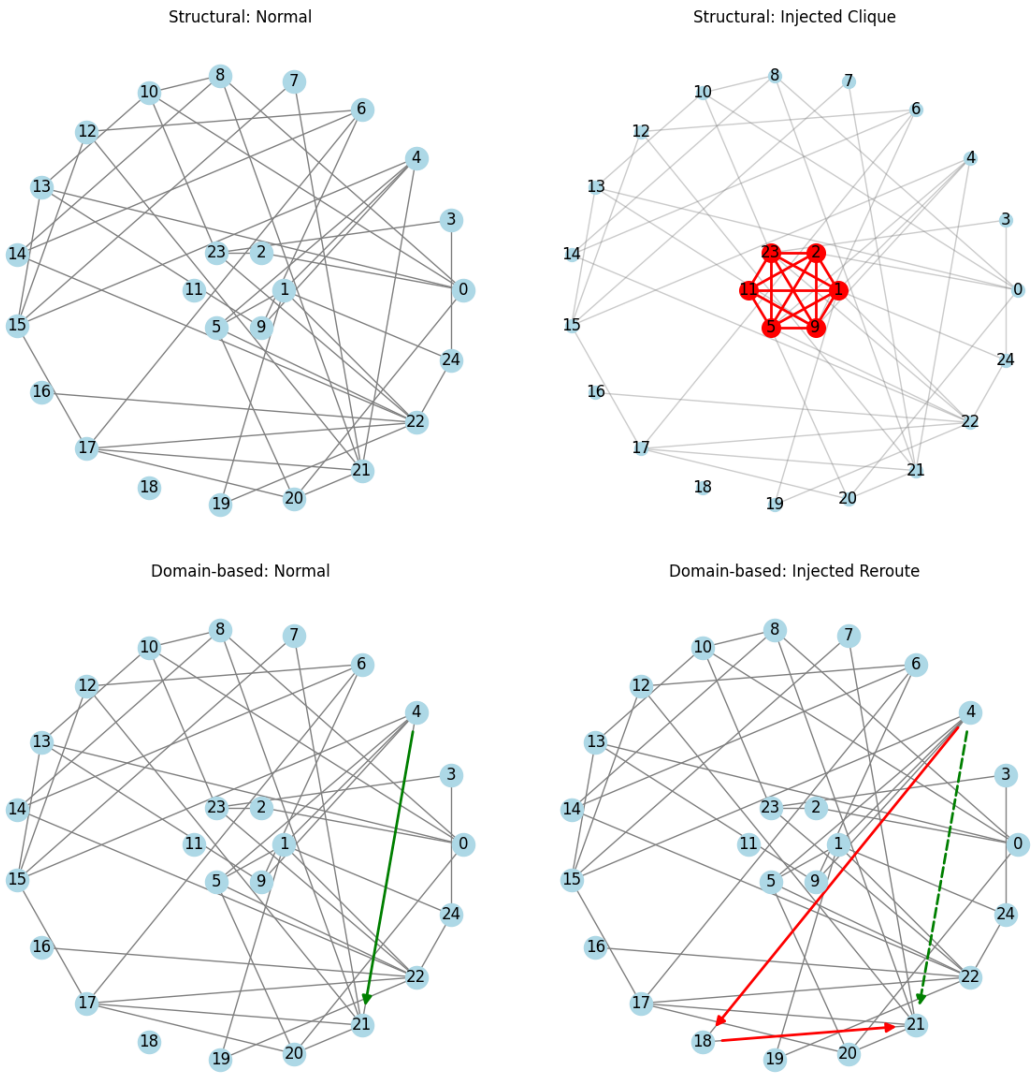


Figure 5.3: Examples of structural and re-routing anomalies.

and testing procedures. The following paragraphs describe the Hyperparameters Optimization, Decoder Architectures and Training Metrics definitions designed for GRAND-Net's training.

5.3.1 Hyperparameter Optimization

Enhancing the model's ability to generate accurate embeddings for financial temporal networks requires selecting feasible ranges for the model's hyperparameters, in this case being the learning rate, weight decay, temporal snapshot interval (number

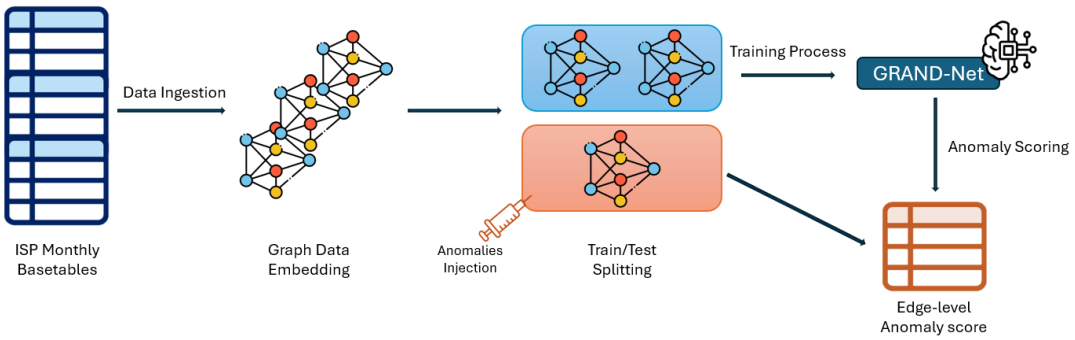


Figure 5.4: GRAND-Net Training and Testing Pipeline.

of days per split) and edge-weight regression decoder type, followed by systematic hyperparameter tuning to identify the optimal configuration.

Hyperparameter tuning was carried out using Optuna's Tree-structured Parzen Estimator (TPE) sampler[2]. Continuous parameters (learning rate, weight decay) were sampled on log-uniform scales; the temporal window length (number of days) was an integer parameter; and the weight-regression decoder was chosen from a set of architectures. Trials were evaluated on validation-set performance, and the best configuration was used for final training.

Table 5.1: Hyperparameter Search Space

Hyperparameter	**Type**	**Search Space**
Learning rate	continuous	LogUniform(10^{-5}, 10^{-2})
Weight decay	continuous	LogUniform(10^{-6}, 10^{-3})
Number of days	integer	IntUniform(1, 30)
Decoder type	categorical	`{SimpleMLP, ResidualMLP, EdgeConv, EdgeAttention, SEAL, TransformerEdge, ResidualEdgeAttention}`

Decoder Architectures :The **SimpleMLPDecoder** concatenates source and target node embeddings and feeds them through a two-layer MLP with ReLU activation. The **ResidualMLPDecoder** augments this with LayerNorm, dropout, and a residual skip connection between hidden layers for better gradient flow and regularization. The **EdgeConvDecoder** follows the EdgeConv paradigm

[2] `https://optuna.readthedocs.io/en/stable/`

by combining the difference $(h_j - h_i)$ with the source embedding h_i in an MLP, capturing local geometric relationships. The **EdgeAttentionDecoder** implements multi-head attention by computing per-head energies on concatenated embeddings, applying a softmax over heads, and merging head-wise features via an MLP. The **TransformerEdgeDecoder** uses each edge's concatenated embedding as a query in a scaled dot-product attention over all node embeddings, followed by an MLP. Finally, the **ResidualEdgeAttentionDecoder** extends multi-head edge-attention with LayerNorm, dropout, and a linear skip-connection on the concatenated attended features before regression.

Results indicated optimal learning rates in the range 1×10^{-4} to 5×10^{-4} and a weight decay of 1×10^{-5}. Temporal window lengths between 2 and 12 days yielded balanced graph densities—shorter windows produced overly sparse graphs, while longer windows diminished temporal resolution. For the decoder choice, the best one proved to be the ResidualEdgeAttentionDecoder: it is the most complex one, but still retains a more than acceptable dimension in terms of model complexity, so it can be safely implemented. Based on validation performance, the final hyperparameters were set as shown in Table 5.2.

Table 5.2: Final Hyperparameter Settings

Hyperparameter	**Final Value**
Learning rate	3×10^{-4}
Weight decay	1×10^{-5}
Number of days	3
Decoder type	**ResidualEdgeAttentionDecoder**

5.3.2 Training and Evaluation Metrics

The model is trained by minimizing a composite loss

$$\mathcal{L} \;=\; \lambda_{\mathrm{LP}}\,\mathcal{L}_{\mathrm{LP}} \;+\; \lambda_{\mathrm{WR}}\,\mathcal{L}_{\mathrm{WR}},$$

where for each edge $(i, j) \in \mathcal{E}$ the model outputs a logit $\hat{s}_{ij} \in \mathbb{R}$ predicting the presence of the edge and a real-valued weight $\hat{w}_{ij} \in \mathbb{R}$, while $y_{ij} \in \{0,1\}$ and $w_{ij} \in \mathbb{R}$ are the corresponding ground-truth label and observed weight. The two loss components are

$$\mathcal{L}_{\mathrm{LP}} = \frac{1}{|\mathcal{E}|} \sum_{(i,j)\in\mathcal{E}} \mathrm{BCEWithLogitsLoss}\Big(\hat{s}_{ij},\, y_{ij}\Big),$$

$$\mathcal{L}_{\mathrm{WR}} = \frac{1}{|\mathcal{E}|} \sum_{(i,j)\in\mathcal{E}} \mathrm{SmoothL1Loss}_{\beta=1.0}\Big(\hat{w}_{ij},\, w_{ij}\Big).$$

Here

$$\text{BCEWithLogitsLoss}(\hat{s}, y) = -\Big[y \log \sigma(\hat{s}) + (1-y) \log\Big(1 - \sigma(\hat{s})\Big)\Big],$$

and for error $e = \hat{w} - w$,

$$\text{SmoothL1}_\beta(e) = \begin{cases} \frac{e^2}{2\beta}, & |e| < \beta, \\ |e| - \frac{\beta}{2}, & |e| \geq \beta. \end{cases}$$

Using this two-coefficients loss system allows the model to define separate evaluation metrics for edge existence prediction and edge weight regression, and to update the loss considering each one as a different heads performing different tasks, each one having its own weight inside the model. The mixing coefficients λ_{LP} and λ_{WR} are indeed not fixed but adaptively optimized during training using a gradient-based class activation mapping (Grad-CAM) approach proposed by [66]: at each step, gradients of the composite loss with respect to each head's output are aggregated and normalized to re-weight the two objectives, allowing the model to emphasize the head providing the stronger learning signal.

In validation and testing the link-prediction quality is measured by the area under the ROC curve AUC_{LP}, regression accuracy by mean-absolute-error

$$\text{MAE} = \frac{1}{|\mathcal{E}|} \sum_{(i,j)\in\mathcal{E}} \Big|\hat{w}_{ij} - w_{ij}\Big|,$$

and downstream anomaly-detection performance by ROC AUC on binary edge-anomaly labels AUC_{AD}.

5.4 Experiments

In this section, the results of our experiments across different model configurations are presented in two parts: (i) encoder comparison in terms of ROC-AUC on the link-prediction task and MAE on the weight-regression task and (ii) anomaly-detection performance, reporting ROC-AUC and class-separability on test graphs with varying anomaly-injection rates. All per-epoch training-loss curves are provided in Appendix II.

5.4.1 Dataset Description

The dataset comprises approximately 60 million wire-transfer transactions over a one-month period from the ISP archives. Each transaction is modeled as an edge in a temporal graph, while nodes are defined by an aggregation key. In this study the three selected keys are:

- **BIC4**: the first four characters of the SWIFT/BIC code, i.e. the institution code, grouping all branches of the same bank together.

- **BIC6**: the first six characters of the SWIFT/BIC code, i.e. institution and country code, grouping banks by both institution and country.

- **Country**: the ISO 3166-1 alpha-2 country code (positions 5–6 of the BIC), aggregating all banks within the same country.

Alternative aggregation keys (e.g. clients IBAN code, or different combinations of BIC codes' characters) yield coarser or finer graph resolutions but use the same downstream code and architecture. Transactions are grouped into temporal windows of fixed length (determined in the Hyperparameters section), and within each window all edges between the same pair of nodes are aggregated by summing their amounts. The aggregated edge weights then serve as edge-attribute inputs to the model. Node features are constructed as time-series vectors recording, for each node and each of the previous T timesteps, its total incoming and total outgoing transaction amounts. This representation jointly captures spatial structure (via the graph) and temporal dynamics (via node feature sequences).

The final size of training, validation and test data is shown in Tables 5.3, 5.4 and 5.5:

Table 5.3: BIC6 Dataset Sizes for Training, Validation and Testing

Dataset	Nodes	Edges	Time Steps	Features
Training	1603	21,587	10	4
Validation	1603	3,084	10	4
Testing	1603	6,168	10	4

Table 5.4: Country Dataset Sizes for Training, Validation and Testing

Dataset	Nodes	Edges	Time Steps	Features
Training	138	5,603	10	4
Validation	138	800	10	4
Testing	138	1,601	10	4

Table 5.5: BIC4 Dataset Sizes for Training, Validation and Testing

Dataset	Nodes	Edges	Time Steps	Features
Training	817	17,232	10	4
Validation	117	2,462	10	4
Testing	233	4,923	10	4

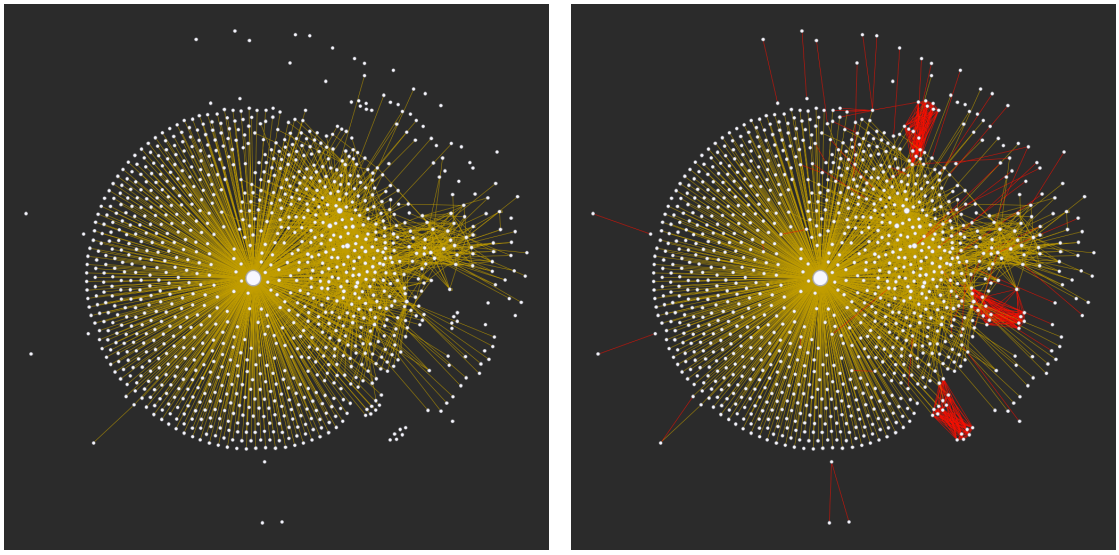


Figure 5.5: Gephi Visualization of one of the network timestamps used in evaluation: the left graph is the real test network, the right one is after anomalies injection (edges in red).

5.4.2 Graph Sampling Algorithms Comparison

As part of the data preprocessing pipeline of the model, graph-downsamplint algorithms listed in Chapter 4 are evaluated and their performance compared. The fidelity of the reconstructed graph to the original one is measured using some properties of the graphs. These properties can take either a distributional form (e.g., degree distribution or distance distribution) or a scalar form obtained by applying an aggregate function (e.g., average degree or diameter) to the said distributions. In the present study, we focus on the following three distributional features:

- *Degree distributions* of nodes (both in-degree and out-degree),
- *Edge-weight distribution* (transactional amounts),

- *Edge-time-occurrence distribution* (timestamps of edge events).

By selecting these three distributions, we aim to ensure that both structural and quantitative characteristics of the original graph are preserved in the sampled subgraph. To evaluate the performance of each sampling algorithm with respect to these distributions, three well-known and robust distributions distance evaluation metrics are employed: the Kolmogorov–Smirnov statistic, the Kullback–Leibler divergence and the Jensen–Shannon divergence. Each sampling algorithm is evaluated via the Kolmogorov–Smirnov, Kullback–Leibler and Jensen–Shannon divergences (D_{KS}, D_{KL}, D_{JS}) on degree, edge-weight and edge-time distributions, where lower values denote higher fidelity. Four target sizes (0.5%, 1%, 5% and 10% of original nodes for vertex methods or edges for edge methods) are tested across ten runs with distinct random seeds. Results of sampling experiments for degree, edges weights and transactions timestamps metrics are shown in Fig. 5.6, 5.7 and 5.8.

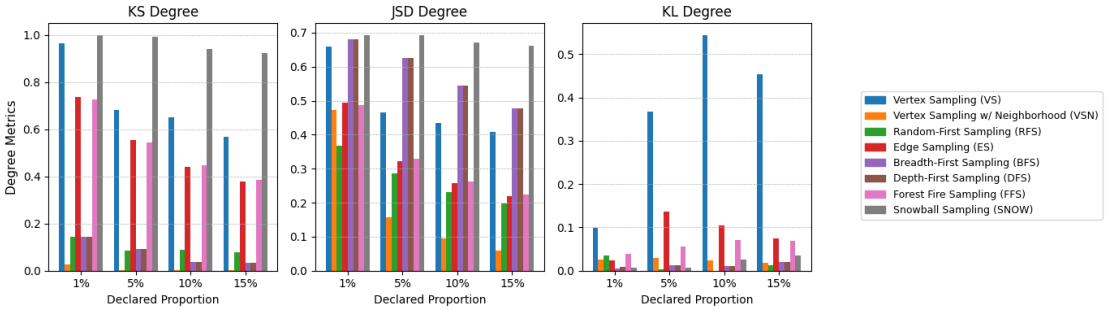


Figure 5.6: Distance metrics for Nodes Degree distributions.

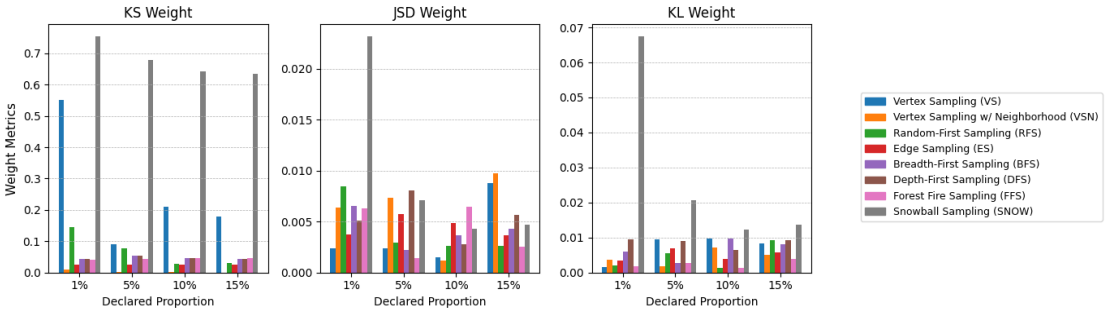


Figure 5.7: Distance metrics for Edges Weight distributions.

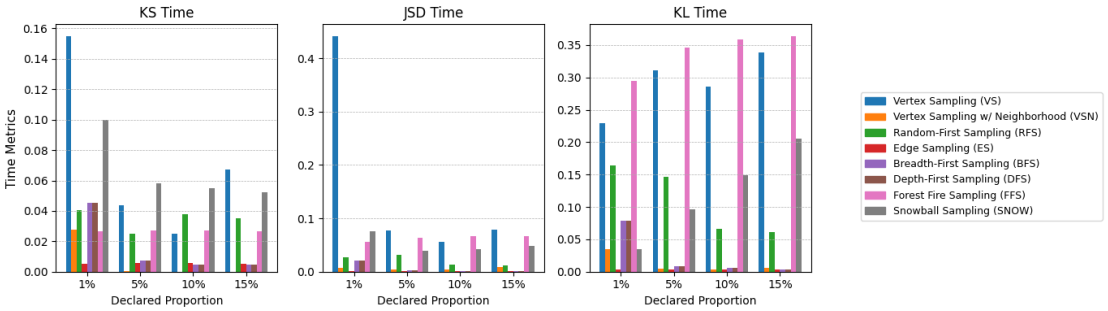


Figure 5.8: Distance metrics for Edges Timestamps distributions.

Results shows that, as expected, VS is trivial and fails to preserve structure ($KS_{deg} \approx 0.964$ at 1%). SNOW/VSN tends to violate node/edge budgets and is really dependent on seeds; BFS/DFS achieve degree fidelity ($KS_{deg} \approx 0.037$ at 10%) but retrieve ∼79% of edges; ES/FFS match edge targets but require ∼45–50% of nodes. In contrast, RFS best balances both: at 1% nodes it retains 0.154% of edges with $KS_{deg} = 0.1429$, $KS_{weight} = 0.1439$; at 5% nodes it retains 6.23% of edges with $KS_{deg} = 0.0857$, $KS_{weight} = 0.0783$, achieving KS< 0.10 as suggested in [60]. The ANNs models proposed in the following sections will be trained on a version of the original data sampled using RFS at 10% of original nodes, and then tested on the real, fully-sized network, demonstrating the capability of this algorithm to produce a lighter and yet representative version of graph data.

5.4.3 Baselines

To evaluate the proposed model's performance, three network embedding based baselines are selected:

- **DeepWalk** [67]: an unsupervised node embedding method that generates truncated random walks over the graph and applies the Skip-Gram model to learn low-dimensional representations capturing local neighborhood structure.
- **Node2Vec** [68]: extends DeepWalk by using a biased random-walk strategy (with return and in-out hyperparameters) to interpolate between breadth-first and depth-first graph explorations, yielding embeddings that balance community and structural equivalence.
- **Spectral Clustering** [69]: applies k-means clustering to the eigenvectors of the graph Laplacian, grouping nodes into clusters based on the spectrum of the Laplacian matrix; To preserve the local connection relationship, the spectral embedding generates the node embedding by maximizing the similarity between nodes in the neighborhood.

5.4.4 Encoder Comparison

To assess encoder performance, each model is trained on the subset of train temporal graphs, each one sampled using RFS. Each encoder is then evaluated on the corresponding test subset. For each temporal graph, the encoder receives as input the actual set of graph edges, considered as positive samples, along with a set of negative samples composed of non-existent edges. A common approach for generating such negative samples involves drawing from a *context-independent* noise distribution, such as random sampling or injected sampling [70]. In this method, negative samples are drawn independently and without reference to the structure of observed data. However, given the vast space of potential anomalous edges, the resulting noise distribution may differ significantly from the empirical data distribution, which can impair model training and generalization.

To mitigate this issue, a *context-dependent* negative sampling strategy is adopted. The core intuition is to generate negative samples that are informed by the graph's structure. Formally, the context-dependent noise distribution for a sampled edge set E_0 is defined as:

$$P_{E_0} \sim P(E) \cdot \left(\frac{1}{N \cdot |E|} \right)$$

where $P(E)$ denotes the distribution of observed edges, $|E|$ is the total number of edges in the graph, and N is the number of nodes.

The procedure begins by randomly selecting a valid edge $e = \{x_a, x_b\}$ from the graph. One of the nodes, say x_a, is then replaced with a randomly sampled node x'_a from the node set, forming a new edge candidate $e' = \{x'_a, x_b\}$. If e' does not exist in the original graph, it is retained as a valid negative sample; otherwise, it is discarded.

Starting from the combined set of positive and negative edges, the network outputs both an existence probability and a predicted weight for each edge. By assigning a weight prediction also to non-existent edges—whose ground truth weight is zero—the model is encouraged to align low existence probabilities with low predicted weights. This dual prediction mechanism enhances the network's ability to learn a coherent representation of the graph structure, ultimately improving the reconstruction of the graph at timestep T.

Table 5.6 and Fig 5.9 summarizes link-prediction AUC and weight-regression MAE for six temporal graph encoders on BIC4, BIC6, and Country aggregations. All models achieve strong performance—AUCs above 0.84 and MAEs below 0.22—demonstrating that graph-based recurrent architectures can effectively capture the structural and temporal dynamics of large-scale wire-transfer networks.

TGCN stands out in link prediction, with AUCs of 0.9894 (BIC4), 0.9890 (BIC6), and 0.9595 (Country), reflecting its superior ability to model inter-node dependencies. In contrast, **DyGR** excels at weight regression, achieving the lowest

MAEs of 0.1328 (BIC4), 0.1327 (BIC6), and 0.1358 (Country), underscoring its precision in estimating transaction amounts. The **GConv–GRU** and **GConv–LSTM** variants offer a balanced trade-off, delivering solid AUCs (≈0.97–0.98 on BIC4/BIC6, ≈0.94 on Country) alongside moderate regression errors (MAE≈0.14–0.16). Both **GCLSTM** and **LRGCN** lag behind on both tasks, with GCLSTM in particular exhibiting elevated MAE values.

Further experiments performed in this work on the downstream Anomaly Detection task will serve to determine which of these strengths—high link-prediction accuracy or low regression error—most effectively translates into identifying anomalous flows of money.

Method	**BIC4**		**BIC6**		**Country**	
	AUC	MAE	AUC	MAE	AUC	MAE
TGCN	**0.9894**	0.1354	**0.9890**	0.1369	**0.9595**	0.1459
GConv–GRU	0.9733	0.1389	0.9723	0.1422	0.9400	0.1455
GConv–LSTM	0.9740	0.1492	0.9731	0.1514	0.9406	0.1578
GCLSTM	0.9427	0.2094	0.9345	0.2129	0.9323	0.2121
LRGCN	0.8443	0.1573	0.8261	0.1636	0.8592	0.1551
DyGR	0.9315	**0.1328**	0.9230	**0.1327**	0.9253	**0.1358**

Table 5.6: Link-prediction AUC and weight-regression MAE of GNN models on datasets aggregated by BIC4, BIC6, and Country (best values in bold).

5.4.5 Anomaly Detection Sensitivity

Model performance on the downstream task of Anomaly Detection is assessed by applying the pre-trained encoders to anomaly-injected subgraphs extracted from the test set. This transforms the problem into binary classification: each edge is assigned by the model a score computed as a weighted sum of its existence probability and predicted weight, and ROC-AUC is evaluated against ground-truth labels (0 for anomalous edges, 1 for normal ones). Figure 5.10 and Table 5.7 illustrates the variation in anomaly-detection ROC-AUC for each model as the fraction of injected anomalies increases from 1% to 20%.

Results show a significant improvement upon static embedding baselines (DeepWalk, Node2Vec, Spectral Clustering) for temporal GNN encoders across all aggregation levels and anomaly-injection rates. The baselines are confined to ROC-AUC values below 0.76 (e.g., Spectral Clustering peaks at 0.7580 on BIC6 and 0.7416 on

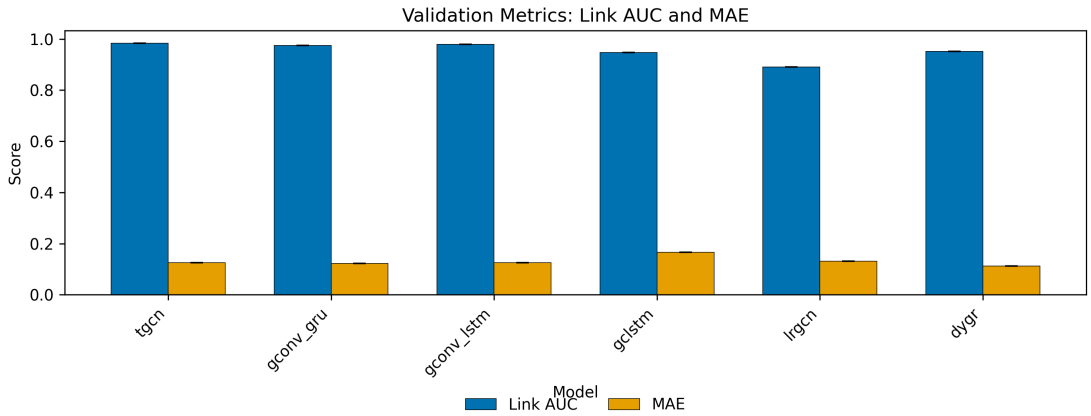


Figure 5.9: Comparison of link-prediction ROC-AUC, regression MAE, and anomaly-detection ROC-AUC (1% injection) across encoders (averaged over the three datasets and over 10 seeds).

Country), whereas every GNN model consistently exceeds 0.88, demonstrating a clear advantage of graph-based recurrent architectures for anomaly detection.

Among the GNNs, **DyGR** attains the highest AUC on both BIC6 (0.9416 at 1% injection) and BIC4 (0.9450 at 1%), while **TGCN** leads on the Country aggregation (0.8800 at 1%, rising to 0.9071 at 5%), highlighting its robustness to varying anomaly rates. The performance drop for DyGR from 1% to 20% injection is only 0.0132 on BIC6 and 0.0103 on BIC4, and similarly modest for TGCN ($\Delta\text{AUC} < 0.01$ on Country), indicating strong stability under increasing anomaly prevalence.

Intermediate models (**GConv–GRU**, **GConv–LSTM**) deliver solid but lower AUCs (approx. 0.92–0.93 on BIC6/BIC4, approx. 0.85–0.88 on Country), while **GCLSTM** and **LRGCN** trail behind (AUCs as low as 0.6467 and 0.8521 at 20% on BIC6, respectively).

On top of the quantitative analysis, it is also possible to visualize class separation for each model by plotting the anomaly score histograms at the 1% injection rate. These plots highlight the distributions of scores assigned to normal edges and the three types of anomalous edges introduced: structural, contextual, and re-routing. In general, most models exhibit good separation between normal and anomalous classes. An exception is observed in the case of the GCLSTM model, which fails to outperform the static baselines, suggesting poor discriminative capability in this setting.

Among the anomaly types, structural and re-routing anomalies consistently

Method	BIC6				BIC4				Country			
	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
DeepWalk	0.6594	0.6506	0.6525	0.6292	0.5655	0.6070	0.5876	0.5973	0.4847	0.6380	0.6983	0.6234
Node2Vec	0.6248	0.5900	0.6156	0.6164	0.6565	0.6465	0.5840	0.5873	0.7270	0.5535	0.5966	0.5292
Spectral Clustering	0.7580	0.6614	0.6935	0.6877	0.6278	0.7118	0.6810	0.6830	0.6758	0.7416	0.7227	0.6641
TGCN	0.9383	0.9324	0.9291	**0.9287**	0.9310	0.9278	0.9248	0.9297	**0.8800**	**0.9071**	**0.9027**	**0.9065**
GConv_GRU	0.9286	0.9278	0.9247	0.9239	0.9291	0.9237	0.9206	0.9269	0.8516	0.8792	0.8686	0.8766
GConv_LSTM	0.9016	0.8986	0.8952	0.8939	0.9118	0.8959	0.8925	0.9018	0.8368	0.8696	0.8560	0.8666
GCLSTM	0.6690	0.6605	0.6560	0.6467	0.6951	0.6702	0.6566	0.6742	0.5671	0.5709	0.5259	0.5690
LRGCN	0.8782	0.8629	0.8563	0.8521	0.8701	0.8620	0.8570	0.8675	0.7556	0.7635	0.7382	0.7660
DyGR	**0.9416**	**0.9339**	**0.9313**	0.9284	**0.9450**	**0.9288**	**0.9270**	**0.9347**	0.8345	0.8546	0.8414	0.8620

Table 5.7: ROC-AUC of baseline and GNN models on datasets aggregated by BIC6, BIC4, and Country under varying anomaly-injection rates. Highest AUC per column is highlighted in bold.

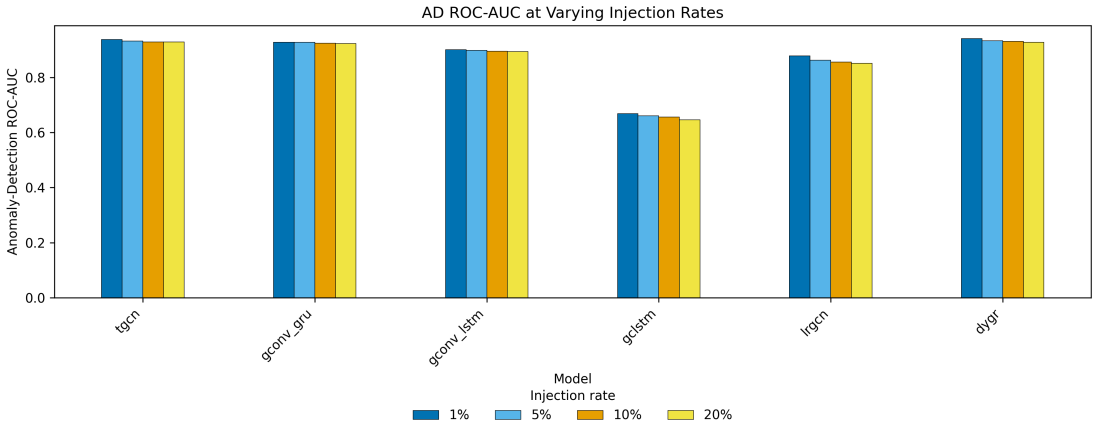
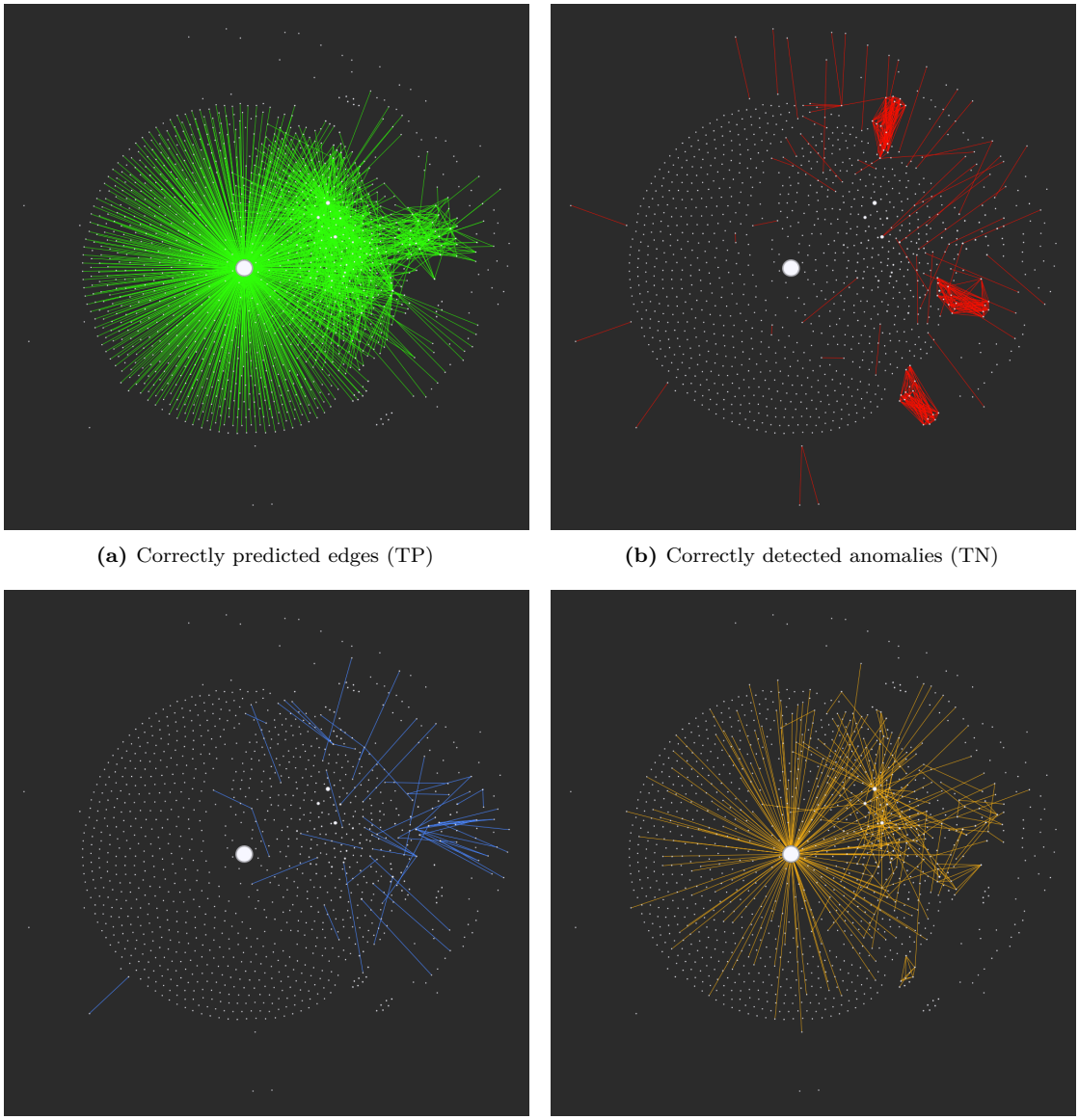


Figure 5.10: Comparison of Anomaly Detection AUC at different injection thresholds across encoders (averaged over the three datasets and over 10 seeds).

emerge as the most difficult to detect, typically resulting in lower anomaly scores. This behavior can be attributed to the inherent nature of these perturbations. Unlike contextual anomalies—which involve modifying the weights of existing edges and therefore only affect the regression head—structural and re-routing anomalies introduce entirely new edges. As a result, they impact both prediction heads: the regression head (since the ground truth weight is zero) and the link prediction head (since these edges are not part of the original, non-perturbed graph and should ideally be classified as non-existent). This dual challenge makes these anomalies particularly difficult for the models to learn and correctly identify.



(a) Correctly predicted edges (TP)

(b) Correctly detected anomalies (TN)

(c) Missed anomalies (FN)

(d) Wrongly predicted edges (FP)

Figure 5.11: An example of predicted network created by GRAND-Net: the overall structure of the ground-truth graph is predicted correctly, with an excellent capability of detecting injected anomalies (see the clear clique structures in figure b).

5.5 Model Deployment on ISP Systems

The model has been designed for seamless integration into the existing ISP AFC/-MAS Transaction Monitoring workflow. It consumes the monthly transaction basetables already produced for other AML processes with virtually no additional

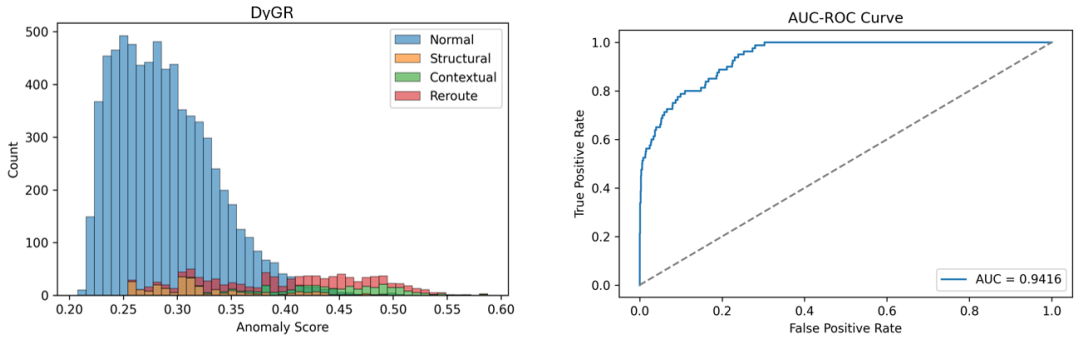


Figure 5.12: Anomalies scores distributions and ROC-AUC curve at 1% injection rate for DyGR model

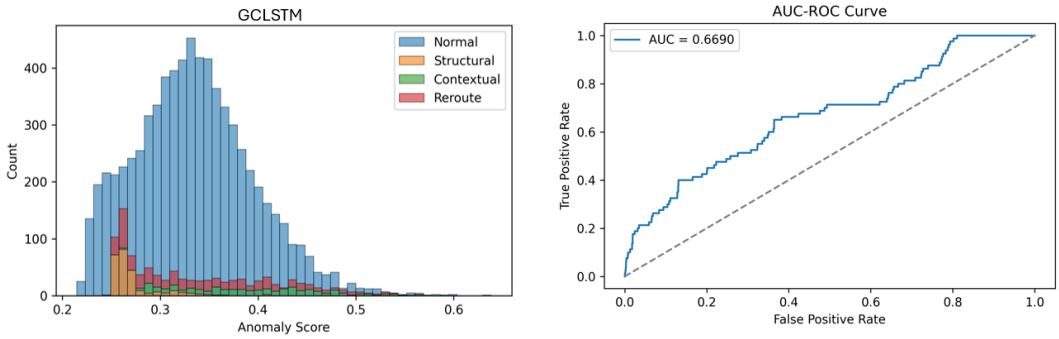


Figure 5.13: Anomalies scores distributions and ROC-AUC curve at 1% injection rate for GCLSTM model

preprocessing. After selecting an encoder of choice, training can be carried out on a rolling window of the desired amount of months (ideally, 24–36 months of historical data, sufficient to capture both long-term trends and seasonal effects) using GPU-equipped workstations provisioned on Google Cloud Platform. Once trained, the model can be executed each time a new basetable is generated by the office systems, producing an anomaly score for every edge of the graph induced by the basetable. Entities whose scores exceed a configurable threshold can be reported as alerts—targeting potential money-laundering, terrorism-financing, sanctions-evasion and related risks - for human operators in the office. Once integrated in the office framework, this pipeline will enable continuous, AI-driven surveillance of the bank's entire transaction stream, enhancing human analysts' ability to detect - and potentially escalate to next offices in the TXM pipeline - suspicious activity in near real time.

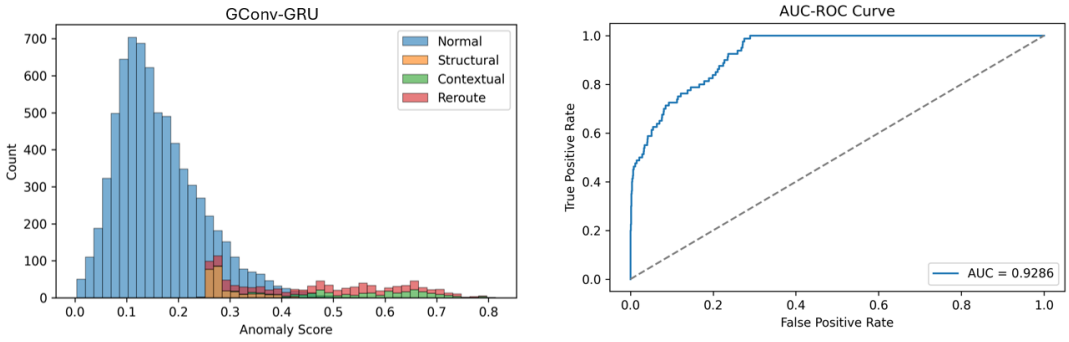


Figure 5.14: Anomalies scores distributions and ROC-AUC curve at 1% injection rate for GConv-GRU model

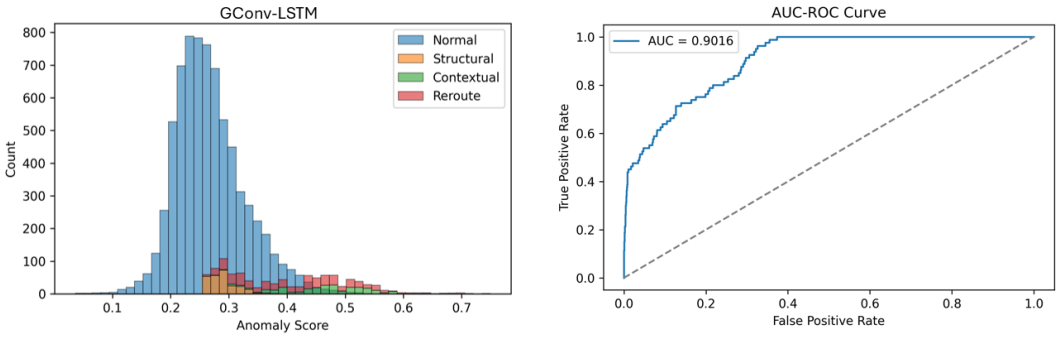


Figure 5.15: Anomalies scores distributions and ROC-AUC curve at 1% injection rate for GConv-LSTM model

5.6 Conclusions

The experiments conducted in this chapter demonstrate that Artificial Neural Network models are a suitable tool to perform anomaly detection tasks on the large-scale financial flows managed by Intesa Sanpaolo. In particular, the proposed model GRAND-Net outperforms pre-existing methods by almost %10 in accuracy, combining structural graph-based encoders and a double classification head to predict anomaly scores for transactional temporal networks. Leveraging the selected downsampling algorithm, the model is able to train on a smaller subset of the original data, improving time and memory-related performances. As with the first model introduced in this thesis, GRAND-Net is already deployed within Intesa Sanpaolo's TXM pipeline. Leveraging the bank's high-performance infrastructure, it can be trained on extended time windows of transactional data and, at predetermined intervals, can be employed to assign anomaly scores to all managed entities at the

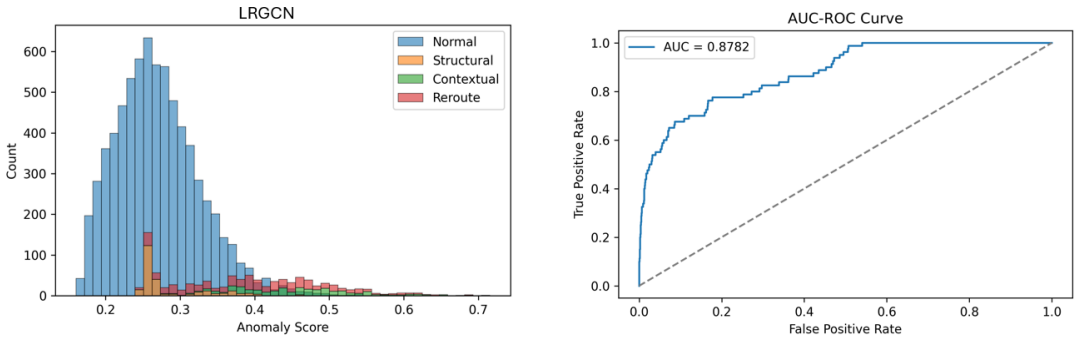


Figure 5.16: Anomalies scores distributions and ROC-AUC curve at 1% injection rate for LRGCN model

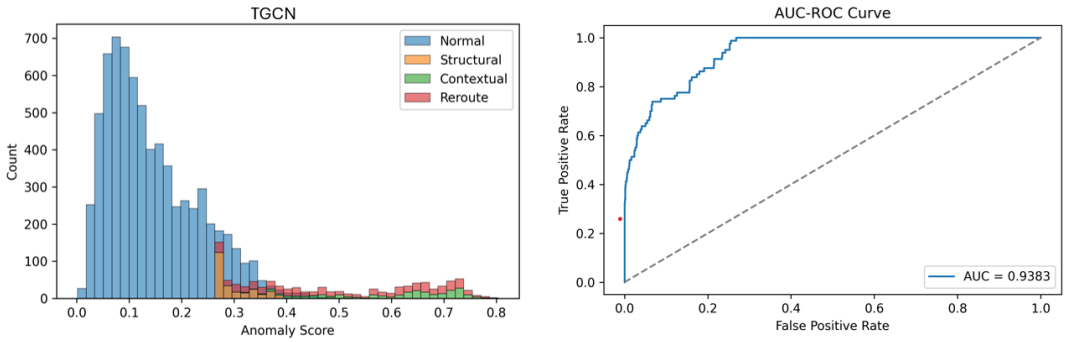


Figure 5.17: Anomalies scores distributions and ROC-AUC curve at 1% injection rate for TGCN model

desired level of aggregation, based on their observed behaviour over the training period. This system will add a novel, AI-based tool to the TXM framework of the bank, providing an innovative top-bottom analysis approach to AML/CFT operations.

Chapter 6

Conclusions and Directions of Research

6.1 Conclusions

This work has introduced TRACED and GRAND-Net: two *network analysis* based solutions to strengthen Intesa Sanpaolo's Transaction Monitoring (TXM) framework. Unlike conventional TXM approaches—which typically treat transactions as isolated tabular entries—both proposed algorithms model financial flows as graphs, capturing complex relational and structural patterns. The first algorithm implements a classic bottom-up pipeline: it aggregates all transactions related to a designated target, applies machine-learning techniques (e.g. community detection on the target's ego-graph), and visualizes the temporal-relational network through a novel "flow-view" interface. This graph-centric visualization enables analysts to trace suspected money-laundering and terrorist-financing patterns more naturally, spotlight the most suspicious counterparties, and follow the directional movement of funds. A final aggregate "suspicion score," computed via network metrics, quantifies each target's overall risk level—facilitating rapid ranking, comparison of clients, and partial automation of the TXM process.

The second algorithm adopts an innovative top-down approach, processing the full monthly transaction data extracted from the bank's core base tables. A Graph-based Recurrent Neural Network takes as input the raw graph spanned by those transactions—where nodes represent clients and edges represent transactions—to flag anomalous edges and nodes that deviate from macro-economic flow patterns. By leveraging both the network's global structure, topological features of each node and the evolution in time of those features, this model uncovers large-scale illicit flow structures—such as hidden layering cycles or indirect fund channels—that remain undetectable by purely local or bottom-up methods.

Both frameworks embody a fully AI-driven, Network-Analysis system trained and validated on proprietary Intesa Sanpaolo transaction data, tailored to the bank's operational requirements, and compliant with the latest AML/CFT regulations and innovation mandates from bodies such as FATF and EUROPOL. By embedding advanced machine-learning, deep-learning, and graph-analysis techniques into a seamless TXM pipeline, these solutions automate detection workflows, reduce false positives, and equip the bank with resilient, adaptive defenses against evolving money-laundering and terrorist-financing threats.

Both models are already operative in the TXM pipeline of Intesa Sanpaolo, and will be live tested in the following months to assess their goodness on real-world scenarios. For this reason, it is impossible at this point of time to predict their exact performance. However, given the solid statistical foundations of the two algorithms and their innovative technological approach to uncovering anomalies among millions of transactions, we anticipate a significant improvement in the current true positive rate of 3% suggested in literature by [11].

6.2 Future Perspectives

Despite the advances presented in this work, the Transaction Monitoring landscape remains full of unexplored or underdeveloped possibilities. A first avenue is the deep integration of *internal intelligence*: by exploiting the richer counterparty metadata enabled by ISO 20022 messaging (see Chapter 2), transaction graphs can be enhanced with relationship hierarchies, risk-profile scores, and customer segmentations to improve model accuracy without relying on external sources. A second avenue is the systematic harvesting and processing of *Open Source Intelligence (OSINT)*—from corporate registries and sanctions databases to news outlets and social media—to inject complementary signals into the monitoring pipeline, albeit at the cost of building robust extraction, linking, and normalization workflows. A third avenue is advanced *Entity Resolution*: applying AI-driven name-matching, ontology-based rules, and sequence models to reconcile ambiguous free-text fields in payment messages (for example "John Green," "J. Green," or "Green John") with actual account holders, ensuring that every node in the transaction graph corresponds to a real-world entity.

Pursuing these directions will not only enrich the network topologies on which the proposed AI-driven models operate—thereby boosting precision and recall in anomaly detection—but also fortify the bank's overall TXM framework. By embedding enhanced intelligence, external context, and resolved entity linkages into existing systems, Intesa Sanpaolo will gain more comprehensive tools to detect suspicious activity, alert authorities, and freeze assets or entities involved in money-laundering and terrorist-financing schemes.

Appendix A

Graph Sampling Pseudocode

VSN Pseudocode

```
1.  Initialize random number generator with given seed.
2.  Sample V_seed <- UniformRandomSubset(V, k).
3.  Initialize V_s <- V_seed.
4.  For each u in V_seed do
5.      Let N_u <- { w in V | (u, w) in E }.
6.      If r is specified then
7.          Sample N_u_sample <- UniformRandomSubset(N_u, min(r, |N_u|)).|
8.      Else
9.          N_u_sample <- N_u.
10.     Add all vertices in N_u_sample to V_s.
11. Initialize E_s <- \emptyset.
12. For each edge (u, v) in E do
13.     If u in V_s and v in V_s then
14.         Add (u, v) to E_s.
15. Return G_s = (V_s, E_s).
```

VS Pseudocode

```
1.  Initialize random number generator with given seed.
2.  Sample V_s <- UniformRandomSubset(V, k).
3.  Initialize E_s <- \emptyset.
4.  For each edge (u, v) in E do
5.      If u in V_s and v in V_s then
6.          Add (u, v) to E_s.
7.  Return G_s = (V_s, E_s).
```

ES Pseudocode

```
1.  Initialize random number generator with given seed.
2.  Sample E_s <- UniformRandomSubset(E, q).
3.  Initialize V_s <- \emptyset.
4.  For each edge (u, v) in E_s do
5.      Add u to V_s.
6.      Add v to V_s.
7.  Return G_s = (V_s, E_s).
```

BFS Pseudocode

```
1.  Initialize random number generator with given seed.
2.  Select a seed vertex v0 <- UniformRandomElement(V).
3.  Initialize V_s <- {v0},        // sampled vertex set
4.              Q   <- [v0],        // FIFO queue
5.  While |V_s| < k and Q is not empty do
6.      u <- Dequeue(Q).
7.      For each neighbor w of u (i.e., (u,w) in E) do
8.          If w not in V_s then
9.              Add w to V_s.
10.             Enqueue(Q, w).
11.             If |V_s| = k then
12.                 Break out of both loops.
13. Initialize E_s <- \emptyset.
14. For each edge (u, v) in E do
15.     If u in V_s and v in V_s then
16.         Add (u, v) to E_s.
17. Return G_s = (V_s, E_s).
```

DFS Pseudocode

```
1.  Initialize random number generator with given seed.
2.  Select a seed vertex v0 <- UniformRandomElement(V).
3.  Initialize V_s <- {v0},       // sampled vertex set
4.              S   <- [v0],       // LIFO stack
5.  While |V_s| < k and S is not empty do
6.      u <- Pop(S).
7.      For each neighbor w of u (i.e., (u, w) in E) do
8.          If w not in V_s then
9.              Add w to V_s.
```

```
10.             Push(S, w).
11.             If |V_s| = k then
12.                 Break out of both loops.
13. Initialize E_s <- \emptyset.
14. For each edge (u, v) in E do
15.     If u in V_s and v in V_s then
16.         Add (u, v) to E_s.
17. Return G_s = (V_s, E_s).
```

FFS Pseudocode

```
1.  Initialize random number generator with given seed.
2.  Select seed vertex v0 <- UniformRandomElement(V).
3.  Initialize V_s <- {v0},        // sampled vertex set
4.             Frontier <- {v0}.
5.  While |V_s| < k and Frontier is not empty do
6.      Initialize NextFrontier <- { }.
7.      For each u in Frontier do
8.          Let U_u <- { w not in V_s | (u, w) in E }.
9.          Draw x_u ~ Geometric(p), capped at |U_u|.
10.         If x_u > 0 and U_u is not empty then
11.             Sample B_u <- UniformRandomSubset(U_u, min(x_u, |U_u|)).|
12.             Add all vertices in B_u to V_s.
13.             Add all vertices in B_u to NextFrontier.
14.      Set Frontier <- NextFrontier.
15.      If Frontier is empty and |V_s| < k then
16.          Select new seed v' <- UniformRandomElement(V \ V_s).
17.          Add v' to V_s and Frontier.
18. Initialize E_s <- \emptyset.
19. For each edge (u, v) in E do
20.     If u in V_s and v in V_s then
21.         Add (u, v) to E_s.
22. Return G_s = (V_s, E_s).
```

SNOW Pseudocode

```
1.  Initialize random number generator with given seed.
2.  Select seed vertex v0 <- UniformRandomElement(V).
3.  Initialize V_s <- {v0},        // sampled vertex set
4.             Frontier <- {v0}.
5.  While |V_s| < k and Frontier is not empty do
6.      Initialize NextFrontier <- { }.
```

```
7.      For each u in Frontier do
8.          Let U_u <- { w not in V_s | (u, w) in E }.
9.          If |U_u| <= r then
10.             B_u <- U_u.
11.         Else
12.             Sample B_u <- UniformRandomSubset(U_u, r).
13.         Add all vertices in B_u to V_s and NextFrontier.
14.      Set Frontier <- NextFrontier.
15.      If Frontier is empty and |V_s| < k then
16.          Select new seed v' <- UniformRandomElement(V \ V_s).
17.          Add v' to V_s and Frontier.
18. Initialize E_s <- \emptyset.
19. For each edge (u, v) in E do
20.     If u in V_s and v in V_s then
21.         Add (u, v) to E_s.
22. Return G_s = (V_s, E_s).
```

RFS Pseudocode

```
1.  Initialize random number generator with given seed.
2.  Select seed vertex v0 <- UniformRandomElement(V).
3.  Initialize V_s <- {v0},          // sampled vertex set
4.             Frontier <- {v0}.
5.  While |V_s| < k and Frontier is not empty do
6.      Select u <- UniformRandomElement(Frontier).
7.      Remove u from Frontier.
8.      Let U_u <- { w not in V_s | (u, w) in E }.
9.      For each w in U_u do
10.         Add w to V_s.
11.         Add w to Frontier.
12.         If |V_s| = k then
13.             Break out of loops.
14.  If Frontier is empty and |V_s| < k then
15.      Select new seed v' <- UniformRandomElement(V \ V_s).
16.      Add v' to V_s and Frontier.
17. Initialize E_s <- \emptyset.
18. For each edge (u, v) in E do
19.     If u in V_s and v in V_s then
20.         Add (u, v) to E_s.
21. Return G_s = (V_s, E_s).
```

RWE Pseudocode

```
1.  Initialize random number generator with given seed.
2.  Select initial vertex v <- UniformRandomElement(V).
3.  Initialize V_s <- {v},          // sampled vertex set
4.  While |V_s| < k do
5.      Draw u ~ Uniform(0,1).
6.      If u < alpha then
7.          v <- UniformRandomElement(V).
8.      Else
9.          Let N_v <- { w | (v, w) in E }.
10.         If N_v is not empty then
11.             Select v <- UniformRandomElement(N_v).
12.     If v not in V_s then
13.         Add v to V_s.
14. Initialize E_s <- \emptyset.
15. For each edge (u, w) in E do
16.     If u in V_s and w in V_s then
17.         Add (u, w) to E_s.
18. Return G_s = (V_s, E_s).
```

Appendix B

Learning Curves

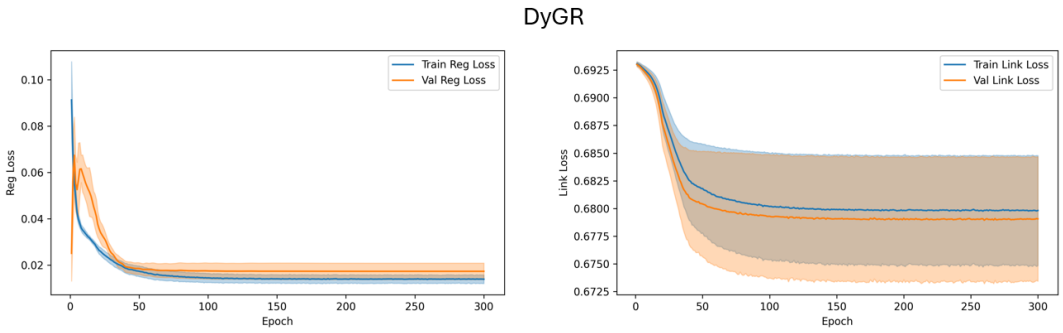


Figure B.1: Learning curves for the link prediction and weight regression heads of DyGR model retrained with 10 different seeds

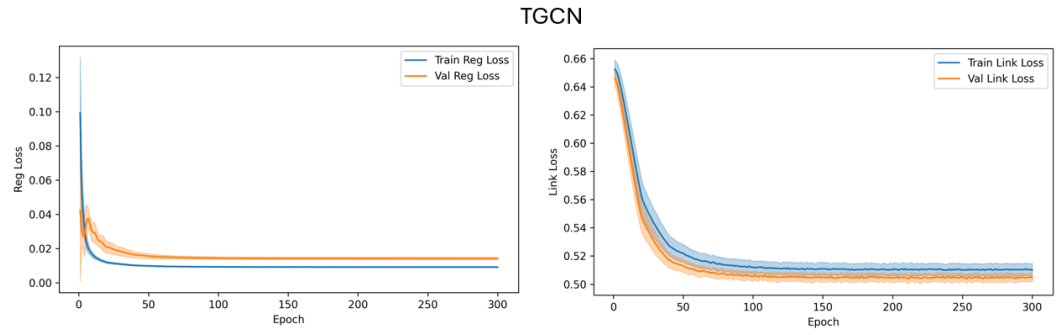


Figure B.2: Learning curves for the link prediction and weight regression heads of TGCN model retrained with 10 different seeds

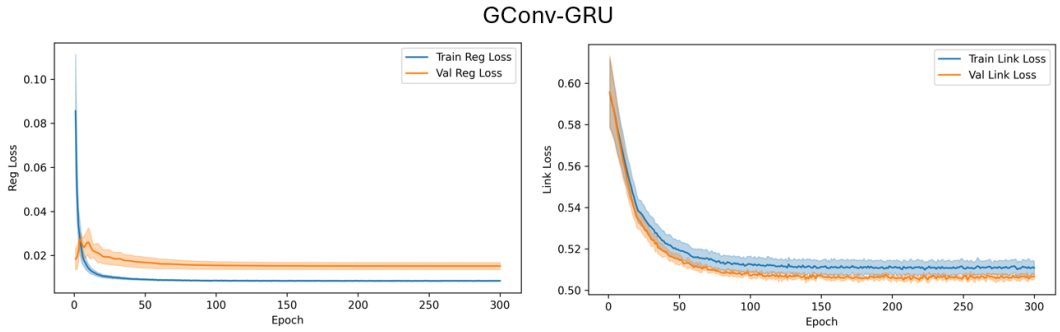


Figure B.3: Learning curves for the link prediction and weight regression heads of GConv-GRU model retrained with 10 different seeds

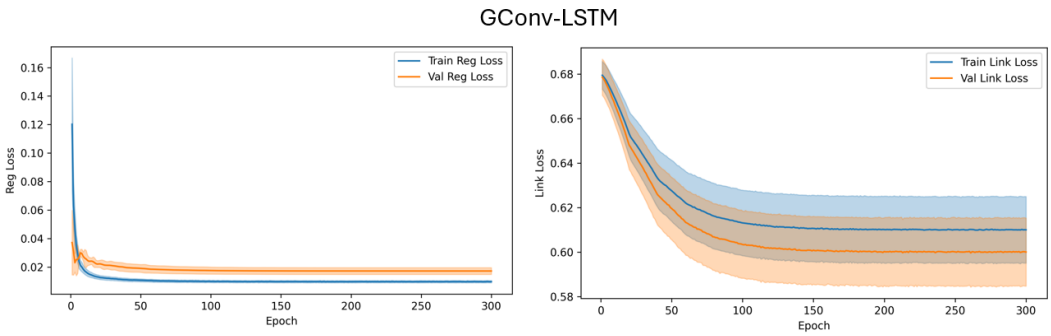


Figure B.4: Learning curves for the link prediction and weight regression heads of GConv-LSTM model retrained with 10 different seeds

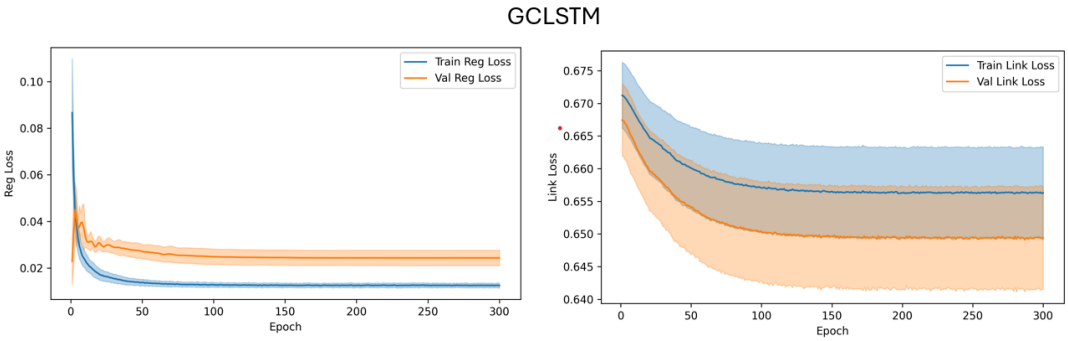


Figure B.5: Learning curves for the link prediction and weight regression heads of GCLSTM model retrained with 10 different seeds

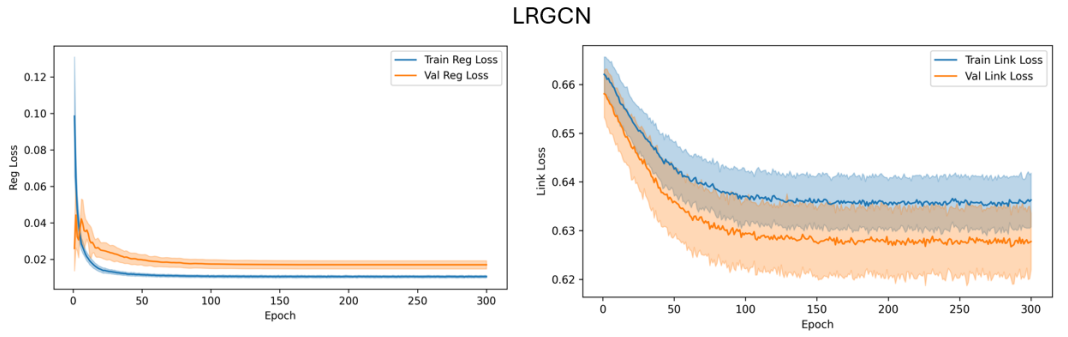


Figure B.6: Learning curves for the link prediction and weight regression heads of LRGCN model retrained with 10 different seeds

Bibliography

[1] John McDowell and Gary Novis. «Consequences of Money Laundering and Financial Crime». In: *Economic Perspectives* 6.2 (May 2001). NCJ No. 191327, p. 1 (cit. on p. 1).

[2] Hugo Almeida, Pedro Pinto, and Ana Fernández Vilas. «A Review on Cryptocurrency Transaction Methods for Money Laundering». In: *arXiv preprint arXiv:2311.17203* (Nov. 2023). arXiv: 2311.17203. URL: https://arxiv.org/abs/2311.17203 (cit. on p. 2).

[3] Financial Action Task Force. *Virtual Assets Red Flag Indicators of Money Laundering and Terrorist Financing*. Tech. rep. Paris, France: Financial Action Task Force, Sept. 2020. URL: https://www.fatf-gafi.org/media/fatf/documents/Virtual-Assets-Red-Flag-Indicators.pdf (cit. on p. 2).

[4] Unità di Informazione Finanziaria per l'Italia. *Rapporto annuale per il 2024, n. 17 – 2025*. Tech. rep. Roma, Italy: Unità di Informazione Finanziaria, Banca d'Italia, 2025. URL: https://uif.bancaditalia.it/pubblicazioni/rapporto-annuale/2025/Rapporto-UIF-anno-2024.pdf (cit. on pp. 2, 4, 7).

[5] Michael Levi and Peter Reuter. «Money laundering». In: *Crime and Justice* (2006) (cit. on p. 2).

[6] A. Bongani Sibindi and W. Gaviyau. «Global Anti-Money Laundering and Combating Terrorism Financing Regulatory Framework: A Critique». In: *Journal of Risk and Financial Management* (2023) (cit. on p. 2).

[7] Antonio Pellegrini, Pierpaolo De Franceschis, Chiara Bentivogli, and Eleonora Laurenza. «Un indicatore sintetico per individuare le società cosiddette cartiere». In: *Quaderni dell'antiriciclaggio, Analisi e studi 15* . (Mar. 2023) (cit. on p. 2).

[8] R. Soltani, U.T. Nguyen, Y. Yang, M. Faghani, A. Yagoub, and A. An. «A new algorithm for money laundering detection based on structural similarity». In: *2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. 2016 (cit. on p. 2).

[9] B. Unger, M. Siegel, J. Ferwerda, W. de Kruijf, M. Busuioic, K. Wokke, and G. Rawlings. *The amounts and the effects of money laundering*. Tech. rep. Report for the Ministry of Finance, 2006 (cit. on p. 2).

[10] United Nations Office on Drugs and Crime. «Money laundering.» In: (). URL: https://www.unodc.org/unodc/en/%20money-laundering/overview.html (cit. on p. 2).

[11] Michele Giammatteo. *N. 26 – Il valore del riciclaggio e delle altre condotte finanziarie illecite in Italia*. Tech. rep. 26. In Italian. Unità di Informazione Finanziaria per l'Italia (Banca d'Italia), Jan. 2025. URL: https://uif.bancaditalia.it/pubblicazioni/quaderni/2025/quaderno-26-2025/index.html (cit. on pp. 3, 22, 76).

[12] Financial Action Task Force. *International Standards on Combating Money Laundering and the Financing of Terrorism & Proliferation: the FATF Recommendations*. Tech. rep. Financial Action Task Force, 2012. URL: https://www.fatf-gafi.org/content/dam/fatf-gafi/recommendations/FATF%20Standards%20-%2040%20Recommendations%20rc.pdf (cit. on pp. 3, 4).

[13] C. Clemente and G. Castaldi. «La normativa in tema di prevenzione del riciclaggio: autorità, regole e controlli». In: *Quaderni dell'antiriciclaggio, Analisi e studi* 20 (Mar. 2023) (cit. on p. 4).

[14] A. Bongani Sibindi and W. Gaviyau. «Global Anti-Money Laundering and Combating Terrorism Financing Regulatory Framework: A Critique». In: *Journal of Risk and Financial Management* (2023) (cit. on p. 5).

[15] Financial Action Task Force (FATF). *FATF Official Website*. https://www.fatf-gafi.org/. 2023 (cit. on p. 5).

[16] G. Pavilidis. «The birth of the new anti-money laundering authority: harnessing the power of EU-wide supervision». In: *Journal of Financial Crime* (2023) (cit. on p. 6).

[17] EBA CLEARING. *EBA CLEARING*. https://www.ebaclearing.eu/. Accessed: June 6, 2025. 2025 (cit. on p. 12).

[18] Rasmus Ingemann Tuffveson Jensen and Alexandros Iosifidis. «Qualifying and raising anti-money laundering alarms with deep learning». In: *Expert Systems with Applications* 214 (Mar. 2023), p. 119037. DOI: 10.1016/j.eswa.2022.119037. URL: https://doi.org/10.1016/j.eswa.2022.119037 (cit. on p. 14).

[19] Annalisa Fronzetti Colladon and Elena Remondi. «Using social network analysis to prevent money laundering». In: *Expert Systems with Applications* 67.1 (2017), pp. 49–58. DOI: 10.1016/j.eswa.2016.09.029. URL: https://doi.org/10.1016/j.eswa.2016.09.029 (cit. on p. 14).

[20] K. Sigetova et al. «Recent trends in the financial crime of the world». In: *Financial and Credit Activity: Problems of Theory and Practice* 5.46 (2022), p. 264. DOI: 10.55643/fcaptp.5.46.2022.3897. URL: https://doi.org/10.55643/fcaptp.5.46.2022.3897 (cit. on p. 14).

[21] M. E. J. Newman. *Networks: An Introduction*. Oxford, UK: Oxford University Press, 2010 (cit. on pp. 15, 18, 44).

[22] Leonhard Euler. *Solutio problematis ad geometriam situs pertinentis*. Published in Commentarii Academiae Scientiarum Imperialis Petropolitanae. 1736. URL: https://scholarlycommons.pacific.edu/euler-works/50/ (cit. on p. 16).

[23] Dénes Kőnig. *Theorie der endlichen und unendlichen Graphen*. First comprehensive book on graph theory. Budapest: Akademiai Kiadó, 1936 (cit. on p. 16).

[24] Jacob L. Moreno. *Who Shall Survive? A New Approach to the Problem of Human Interrelations*. Introduced sociometry and sociograms. Washington, D.C.: Nervous and Mental Disease Publishing Co., 1934 (cit. on p. 16).

[25] Mark S. Granovetter. «The Strength of Weak Ties». In: *American Journal of Sociology* 78.6 (1973). Introduced the concept of weak ties in social networks, pp. 1360–1380 (cit. on p. 16).

[26] Stanley Milgram. «The Small-World Problem». In: *Psychology Today* 1 (1967). Introduced the "six degrees of separation" concept, pp. 61–67 (cit. on p. 16).

[27] Paul Erdős and Alfréd Rényi. «On Random Graphs I». In: *Publicationes Mathematicae* 6 (1959). Introduced random graph theory, pp. 290–297 (cit. on p. 16).

[28] Duncan J. Watts and Steven H. Strogatz. «Collective dynamics of 'small-world' networks». In: *Nature* 393 (1998). Introduced the small-world network model, pp. 440–442 (cit. on p. 17).

[29] Albert-László Barabási and Réka Albert. «Emergence of scaling in random networks». In: *Science* 286.5439 (1999). Introduced the scale-free network model, pp. 509–512 (cit. on p. 17).

[30] Dawei Cheng, Yujia Ye, Sheng Xiang, Zhenwei Ma, Ying Zhang, and Changjun Jiang. «Anti-Money Laundering by Group-Aware Deep Graph Learning». In: *IEEE Transactions on Knowledge and Data Engineering* 35.12 (Dec. 2023), pp. 12444–12457. DOI: 10.1109/TKDE.2023.3272396. URL: https://doi.org/10.1109/TKDE.2023.3272396 (cit. on pp. 17, 19).

[31] Mário Cardoso, Pedro Saleiro, and Pedro Bizarro. «LaundroGraph: Self-Supervised Graph Representation Learning for Anti-Money Laundering». In: *arXiv preprint arXiv:2210.14360* (Oct. 2022). URL: https://arxiv.org/abs/2210.14360 (cit. on p. 18).

[32] Bruno Deprez, Toon Vanderschueren, Bart Baesens, Tim Verdonck, and Wouter Verbeke. «Network Analytics for Anti-Money Laundering – A Systematic Literature Review and Experimental Evaluation». In: (2024) (cit. on pp. 19, 20, 22).

[33] M. Starnini, Ch. Tsourakakis, D. Moncalvo, and others. «Smurf-Based Anti-money Laundering in Time-Evolving Transaction Networks.» In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2021) (cit. on pp. 21, 23).

[34] H. Tariq and M. Hassani. «Topology-Agnostic Detection of Temporal Money Laundering Flows in Billion-Scale Transactions». In: *arXiv preprint arXiv:2309.13662* (2023). Accessed: June 6, 2025 (cit. on pp. 21, 23, 27, 33, 34).

[35] Andrea Fronzetti Colladon and Elisa Remondi. «Using Social Network Analysis to Prevent Money Laundering». In: *Expert Systems with Applications* 67 (2017), pp. 49–58. DOI: 10.1016/j.eswa.2016.09.029. URL: https://www.sciencedirect.com/science/article/pii/S0957417416305139 (cit. on p. 21).

[36] Jianying Xiong and Haifeng Zhong. «Identification of Money Laundering Accounts Based on Weighted Capital Flow Network». In: *Journal of Physics: Conference Series* 1629.1 (Sept. 2020), p. 012023. DOI: 10.1088/1742-6596/1629/1/012023. URL: https://dx.doi.org/10.1088/1742-6596/1629/1/012023 (cit. on p. 21).

[37] Richard J. Bolton and David J. Hand. «Statistical Fraud Detection: A Review». In: *Statistical Science* 17.3 (2002), pp. 235–255. DOI: 10.1214/ss/1042727940. URL: https://doi.org/10.1214/ss/1042727940 (cit. on p. 21).

[38] Leman Akoglu, Hanghang Tong, and Danai Koutra. «Graph Based Anomaly Detection and Description: A Survey». In: *Data Mining and Knowledge Discovery* 29.3 (2015), pp. 626–688. DOI: 10.1007/s10618-014-0365-y. URL: https://doi.org/10.1007/s10618-014-0365-y (cit. on p. 21).

[39] Rafał Drezewski, Jan Sepielak, and Wojciech Filipkowsk. «The application of social network analysis algorithms ˙ in a system supporting money laundering detection.» In: *Information Sciences, 295:18–32* (2015) (cit. on p. 23).

[40] Dawei Zhou, Si Zhang, Mehmet Yigit Yildirim, Scott Alcorn, Hanghang Tong, Hasan Davulcu, and Jingrui He. «A local algorithm for structure-preserving graph cut». In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '17, page 655–664* (2017) (cit. on p. 23).

[41] Xiangfeng Li, Shenghua Liu, Zifeng Li, Xiaotian Han, Chuan Shi, Bryan Hooi, He Huang, and Xueqi Cheng. «Flowscope: Spotting money laundering based on graphs». In: *Proceedings of the AAAI Conference on Artificial Intelligence* (2020) (cit. on p. 23).

[42] Michael Ovelgönne, Chanhyun Kang, Anshul Sawant, and VS Subrahmanian. «Covertness centrality in networks». In: *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pages 863–870* (2012) (cit. on p. 23).

[43] Remco Chang, Mohammad Ghoniem, Robert Kosara, William Ribarsky, Jing Yang, Evan Suma, Caroline Ziemkiewicz, Daniel Kern, and Agus Sudjianto. «Wirevis: Visualization of categorical, time-varying data from financial transactions». In: *2007 IEEE symposium on visual analytics science and technology, pages 155–162* (2007) (cit. on p. 24).

[44] Tat-Man Cheong and Yain-Whar Si. «Event-based approach to money laundering data analysis and visualization.» In: *Proceedings of the 3rd International Symposium on Visual Information Communication* (2010) (cit. on pp. 24, 25).

[45] Financial Action Task Force (FATF). *Guidance on Money Mule Typologies*. Paris: FATF, 2023 (cit. on p. 31).

[46] Financial Action Task Force (FATF). *Jurisdictional Risk Assessment: SWIFT Transfers*. Paris: FATF, 2025 (cit. on p. 32).

[47] Erik Altman, Jovan Blanuša, Luc von Niederhäusern, Béni Egressy, Andreea Anghel, and Kubilay Atasu. «Realistic Synthetic Financial Transactions for Anti-Money Laundering Models». In: *NeurIPS Datasets and Benchmarks Workshop*. arXiv:2306.16424. 2023 (cit. on p. 39).

[48] Haseeb Tariq and Marwan Hassanii. «Topology-Agnostic Detection of Temporal Money Laundering Flows in Billion-Scale Transactions.» In: (2023) (cit. on p. 39).

[49] Frank Rosenblatt. «The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain». In: *Psychological Review* 65.6 (1958), pp. 386–408 (cit. on p. 44).

[50] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. «Learning representations by back-propagating errors». In: *Nature* 323.6088 (1986), pp. 533–536 (cit. on p. 45).

[51] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». In: *Advances in Neural Information Processing Systems*. Vol. 25. 2012, pp. 1097–1105 (cit. on pp. 45, 47).

[52] Othon Michail. «An Introduction to Temporal Graphs: An Algorithmic Perspective». In: *Internet Mathematics* 12 (2016), pp. 239–280. ISSN: 1542-7951. DOI: 10.1080/15427951.2016.1177801. URL: https://doi.org/10.1080/15427951.2016.1177801 (cit. on p. 50).

[53] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. «Structured Sequence Modeling with Graph Convolutional Recurrent Networks». In: *arXiv preprint arXiv:1612.07659* (2016) (cit. on p. 51).

[54] Jinyin Chen, Xueke Wang, and Xuanheng Xu. «GC-LSTM: Graph Convolution Embedded LSTM for Dynamic Link Prediction». In: *arXiv preprint arXiv:1812.04206* (2018) (cit. on p. 51).

[55] Jia Li, Zhichao Han, Hong Cheng, Jiao Su, Pengyun Wang, Jianfeng Zhang, and Lujia Pan. «Predicting Path Failure in Time-Evolving Graphs». In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*. 2019 (cit. on p. 52).

[56] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. «EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs». In: *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*. 2020 (cit. on p. 52).

[57] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. «T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction». In: *arXiv preprint arXiv:1811.05320* (2018) (cit. on p. 52).

[58] E. Kurshan and H. Shen. «Graph Computing for Financial Crime and Fraud Detection: Trends, Challenges and Outlook». In: *arXiv preprint arXiv:2103.03227* (2021). URL: https://arxiv.org/abs/2103.03227 (cit. on p. 52).

[59] Sinno Jialin Pan and Qiang Yang. «A Survey on Transfer Learning». In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359. DOI: 10.1109/TKDE.2009.191 (cit. on p. 52).

[60] Pili Hu and Wing Cheong Lau. «A Survey and Taxonomy of Graph Sampling». In: *CoRR* abs/1308.5865 (2013). URL: http://arxiv.org/abs/1308.5865 (cit. on pp. 53, 54, 66).

[61] Chris Doerr and Norbert Blenn. «Metric Convergence in Social Network Sampling». In: *Proceedings of the 2nd Workshop on Hot Topics in Planet-Scale Measurement (HotPlanet)*. 2013 (cit. on p. 54).

[62] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. «Graphs Over Time: Densification Laws, Shrinking Diameters and Possible Explanations». In: *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '05)*. 2005, pp. 177–187 (cit. on p. 54).

[63] Leo A. Goodman. «Snowball Sampling». In: *The Annals of Mathematical Statistics* 32.1 (1961), pp. 148–170 (cit. on p. 54).

[64] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. «Fast Random Walk with Restart and Its Applications». In: *Proceedings of the Sixth IEEE International Conference on Data Mining (ICDM)*. 2006, pp. 613–622 (cit. on p. 58).

[65] Ming Tu, Jing Huang, Xiaodong He, and Bowen Zhou. «Multiple Instance Learning with Graph Neural Networks». In: *ICML Workshop on Learning and Reasoning with Graph-Structured Representations*. 2019 (cit. on p. 58).

[66] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. «Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization». In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 618–626 (cit. on p. 62).

[67] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. «DeepWalk: Online Learning of Social Representations». In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '14. New York, NY, USA: ACM, 2014, pp. 701–710. DOI: 10.1145/2623330.2623732 (cit. on p. 66).

[68] Aditya Grover and Jure Leskovec. «node2vec: Scalable Feature Learning for Networks». In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, CA, USA: ACM, 2016, pp. 855–864. DOI: 10.1145/2939672.2939754 (cit. on p. 66).

[69] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. «On Spectral Clustering: Analysis and an Algorithm». In: *Advances in Neural Information Processing Systems*. NIPS '02. MIT Press, 2002, pp. 849–856 (cit. on p. 66).

[70] Leman Akoglu, Hanghang Tong, and Danai Koutra. «Graph-based anomaly detection and description: A survey». In: *Data Mining and Knowledge Discovery* 29.3 (2015), pp. 626–688 (cit. on p. 67).