UF-FAE: A Money Laundering Detection System Framework Integrating Union-Find Algorithm and Machine Learning for Distributed Ledger Applications

Zhen Hong[a,*]

[a]*Graduate Institute of Telecommunications and Communication, National Chung Cheng University, Chiayi, Taiwan*

ARTICLE INFO

Keywords:
Anti-Money Laundering (AML)
Union–Find Algorithm
Dynamic Connectivity
Graph-Based Feature Engineering
Multimodal Feature Fusion

ABSTRACT

This study proposes UF-FAE (Union-Find-based Feature-Augmented Embedding), a scalable and theoretically grounded framework for anti-money laundering (AML) detection in large-scale, cross-bank, and highly dynamic transaction networks. The core insight of this work is that many AML detection tasks fundamentally rely on connectivity evaluation and group-merging events, which can be realized using the Union–Find data structure with amortized time complexity O((n)), the lowest known complexity for dynamic connectivity problems. To establish rigorous mathematical foundations, this study further proves that the group partitions maintained by Union–Find are topologically equivalent to the path-connected components of a graph, demonstrating that Union–Find is not merely a data structure but a discrete realization of a fundamental topological invariant.

Built upon this structure, UF-FAE extracts group-level graph-theoretic descriptors (e.g., centrality, reciprocity, bidirectional flow ratio) and integrates them with raw transactional features through a multimodal feature-fusion pipeline implemented in PySpark. The dataset, SAML-D, is processed under strict time-series splitting to prevent future-information leakage. An extensive ablation study evaluates three feature configurations—graph-only, raw-only, and multimodal—combined with four machine-learning classifiers: Logistic Regression, Decision Tree, Random Forest, and Linear SVM.

Experimental results show that graph-based features achieve the strongest detection performance, with Random Forest reaching a Recall of 0.9127 and F1-score of 0.9544, while multimodal fusion further enhances robustness and AUC. The findings confirm that UF-FAE enables near–real-time AML detection, offering both interpretability and scalability. The framework provides a new design paradigm for AML systems, demonstrating that effective detection can be achieved through topological connectivity, dynamic group features, and lightweight learning models, without relying on computationally expensive global graph algorithms.

1. Introduction

The proliferation of digital finance and electronic payment systems has led to increasingly sophisticated patterns of money laundering [1]. Criminal organizations exploit complex transaction structures to disguise illicit funds and integrate them into the legitimate economy. Recent reports estimate that money laundering accounts for approximately 5% of global GDP [2–4]. Despite its complexity, money laundering still follows three canonical stages— placement, layering, and integration [5]—and typically manifests in two dominant flow patterns: scatter–gather and gather–scatter [6, 7]. These characteristics indicate that laundering behaviour remains traceable, especially when analysed using graph-theoretic models of social or transactional networks [8]. Motivated by these observations, this study adopts graph algorithms as the analytical foundation for tracking fund flows.

Blockchain technology further enhances anti-money laundering (AML) capabilities by improving transparency and traceability of transaction flows [9]. However, existing AML systems still rely primarily on static graphs and rule-based filtering, making them inadequate for identifying complex and adaptive laundering strategies [4, 10, 11]. Additionally, regulatory constraints prevent financial institutions from sharing customer-level transactional data, resulting in information silos that hinder cross-institutional tracking [12].

Current machine-learning-based AML approaches often depend heavily on heuristics, suffer from limited interpretability, and show weak generalization across different financial environments [2, 13]. Prior studies emphasize

*Corresponding author
leon28473787@gmail.com (Z. Hong)
ORCID(s):

the importance of network structure in financial anomaly detection [14], and graph-based methods have demonstrated strong effectiveness in identifying abnormal behaviour in large-scale transaction systems [1, 15]. Yet machine-learning systems alone remain computationally expensive and difficult to operate in real-time settings, limiting their practical deployment in high-throughput AML pipelines [16].

To address these challenges, this work proposes **UF-FAE (Union-Find Feature-Augmented Embedding)**, a dynamic graph-driven AML framework that decomposes large transaction networks into weakly connected components using Union–Find with path compression. From each component, graph-theoretic and statistical descriptors are extracted to characterize group behaviour and identify anomalies. The model is further evaluated through time-aware ablation experiments on the SAML-D dataset.

The main contributions of this study are summarized as follows:

1. We propose **UF-FAE**, a real-time, interpretable, and scalable AML graph-grouping framework.
2. We demonstrate the **topological equivalence of Union–Find and connected-component separation** (DSU = π_0), establishing a formal foundation for AML grouping.
3. We construct a **multi-modal feature set** to capture group evolution, structural behaviour, and laundering-related anomalies.
4. We conduct **time-ordered ablation experiments on the SAML-D dataset** to validate the effectiveness and robustness of UF-FAE.

2. Related Work

2.1. Limitations of Machine Learning in AML

Machine learning has been widely applied to anti-money laundering (AML) due to its ability to model complex behavioural patterns [10]. However, AML datasets are notoriously imbalanced: transactions labelled as laundering typically constitute less than 0.1% of all financial activity. This extreme imbalance poses severe challenges for traditional supervised learning models, resulting in high false-negative rates and unstable performance.

Prior studies have systematically highlighted the limitations of machine learning in AML anomaly detection. Study [17] proposed the APATE framework, which integrates customer behavioural features with transaction-network structures, demonstrating significant improvements in credit-card fraud detection. Work [18] emphasized the necessity of combining graph neural networks with machine learning in order to capture the dynamic and relational nature of transaction networks. Study [19] further incorporated extensive feature engineering and XGBoost, achieving competitive performance but acknowledging the high computational cost associated with engineering features over large-scale financial data. Meanwhile, [13] introduced EvolveGCN, which models temporal evolution directly within the GCN parameters; however, deep learning approaches suffer from even higher computational complexity and reduced interpretability, making them difficult to adopt in regulatory and compliance environments.

Given these limitations, this work argues that AML detection must leverage real-world transaction-network structures while maintaining high throughput and operational efficiency. Union–Find, a deterministic algorithm with near-constant-time complexity $O(\alpha(n))$, offers these advantages. When combined with distributed ledger technologies to overcome cross-institutional data silos, Union–Find provides a practical and efficient foundation for large-scale AML system design.

2.2. Applications of Blockchain, Graph Algorithms, and Machine Learning in AML

In large-scale data analysis, identifying anomalies is as important as understanding global structure. Within data mining, the domain specifically focused on detecting rare or irregular patterns is known as anomaly detection [20]. In financial systems, graph-theoretic algorithms are frequently employed to identify suspicious or laundering-related transactions [21, 22].

Study [23] combined transaction-graph features with account-association analysis to uncover money-laundering behaviour, aligning with this work's use of graph structures for AML. Study [24] emphasized that class imbalance remains a major challenge in blockchain-based AML datasets, and proposed the XGBCLUS method, which improves illicit-behaviour scoring without sacrificing critical information. Using SHAP to interpret the ensemble model, their results outperformed single decision-tree classifiers in accuracy and true-positive rates, providing valuable insights for enhancing interpretability in anomaly detection.

Together, [24, 25] explored two critical challenges in AML research: class imbalance and multi-modal fusion. While [24] used XGBCLUS and SHAP to alleviate imbalance and interpret model predictions, [25] constructed a multi-source Ethereum-based feature model incorporating blacklist and phishing datasets, along with explainable analysis. Two conclusions from these studies strongly motivate the UF-FAE design:

1. Multi-modal features significantly outperform single-source features, supporting UF-FAE's integration of raw transaction attributes with graph-theoretic descriptors.
2. Interpretability is essential for AML, reinforcing this study's use of explainable graph indicators such as centrality, group size, and merging events.

Building on these insights, UF-FAE incorporates a topological justification showing that algorithmic group detection is theoretically sound and practically valid for AML. The use of behavioural graph indicators further enhances model accuracy and training efficiency. Consequently, integrating graph algorithms, distributed ledger architectures, and machine learning holds significant promise as a next-generation AML detection paradigm.

This study draws inspiration from IOTA Tangle 2.0, which replaces the traditional linear blockchain with a Directed Acyclic Graph (DAG). Using the OTFPC mechanism, each new transaction verifies two previous ones through a randomized depth-first-search process [26, 27]. While Tangle 2.0 provides high parallelism and verification efficiency, it does not model social-network-like structures. Therefore, this study employs UF-FAE to cluster accounts and transactions using a deterministic Union–Find approach, ensuring interpretability and reproducibility.

DAGs are a core data structure in graph analytics. PageRank [28] demonstrated the power of DAG-based importance scoring and influenced decades of research. For high-efficiency AML detection, practical systems require algorithms that are compatible with parallel computation. Union–Find with path compression offers near-constant amortized time $O(\alpha(n))$ [29, 30], making it ideal for large-scale graph clustering. Later chapters provide a topological proof of this property.

Blockchain systems naturally exhibit distributed characteristics and can be extended to the Bulk Synchronous Parallel (BSP) model, which also forms the basis for distributed implementations of Union–Find. In this study, accounts are treated as nodes and transactions as directed edges. Because Union–Find operates on undirected weakly connected components (WCC), directed edges are first converted into undirected form for clustering. After WCC construction, the graph is then restored to directed form for subsequent analysis. Given that financial data includes explicit temporal ordering, the resulting structure is well suited for DAG-based modelling, enabling Union–Find to efficiently cluster large networks with clear causal constraints.

The overall system concept aims to integrate UF-FAE with BSP-style distributed processing, enabling multi-modal behavioural features to reflect the temporal and structural evolution of transaction communities. The resulting design is compatible with multi-core, cloud, and large-scale computing environments, while satisfying regulatory requirements for interpretability. Subsequent chapters provide detailed explanations of the UF-FAE algorithm, its topological foundations, and its experimental workflow.

3. Problem Formulation

The UF-FAE framework begins by applying Union–Find with path compression and union-by-rank to decompose the transaction graph into multiple weakly connected components (WCCs). This reduces the computational complexity associated with large-scale financial networks. For each resulting subgraph, graph-theoretic indicators and transaction-level statistical features are extracted and assembled into feature vectors for machine learning.

Let the transaction dataset define a directed graph $G = (V, E)$, where each vertex $v \in V = \{a_1, a_2, \dots, a_n\}$ represents an account, and each directed edge $(u, v) \in E$ denotes a transfer from sender u to receiver v. Union–Find with path compression and union-by-rank is then applied to obtain WCCs as follows:

1. Each vertex is initialized with a parent pointer.
2. For every observed transaction (u, v), the sets of u and v are merged.

This process partitions the directed graph G into k weakly connected subgraphs:

$$G_1, G_2, \dots, G_k.$$

Node Role Classification

For each node v in subgraph G_i, in-degree and out-degree are computed:

$$\deg_{\text{in}}(v) = |\{\, u \mid (u,v) \in E \,\}|, \qquad \deg_{\text{out}}(v) = |\{\, w \mid (v,w) \in E \,\}|.$$

Nodes are classified into behavioural roles:

$$\deg_{\text{in}}(v) = 0 \;\Rightarrow\; v \in \text{Sender}, \qquad \deg_{\text{out}}(v) = 0 \;\Rightarrow\; v \in \text{Receiver}.$$

This provides the basis for distinguishing anomalous transactional modes.

Graph-Theoretic Indicators

For each subgraph G_i, the following graph metrics are computed.

Bidirectional Edge Ratio

$$\text{BidirectRatio}(G_i) = \frac{|\{\, (u,v) \in E_i \mid (v,u) \in E_i \,\}|}{|E_i|}.$$

Degree Centrality For each node v:

$$\deg(v) = |\{\, u \mid (u,v) \in E_i \,\}| + |\{\, w \mid (v,w) \in E_i \,\}|.$$

Closeness Centrality To avoid directional connectivity issues, each subgraph is considered as undirected:

$$C_{\text{closeness}}(v) = \frac{1}{\sum_{u \neq v} d(v,u)},$$

where $d(v,u)$ is the shortest-path distance between v and u.

Betweenness Centrality

$$C_{\text{betweenness}}(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}},$$

where σ_{st} is the number of shortest paths from s to t, and $\sigma_{st}(v)$ is the count of those paths passing through v.

Subgraph Structural Statistics

For each G_i, structural descriptors are recorded:

- **Node count** $|V_i|$ and **edge count** $|E_i|$, capturing the local network scale.
- **Bidirectional edge ratio**, reflecting internal interaction density and transaction complexity.
- **Centrality indicators** (closeness, betweenness) computed separately for Senders and Receivers to identify potential hubs or abnormal accounts within each group.

4. Algorithm Design

4.1. Union–Find

The Union–Find (also known as Disjoint Set Union, DSU) data structure is a classical method for maintaining connectivity relationships among elements. Its origins trace back to the early set-union algorithms in [31], with major optimizations introduced in [29], including path compression and union-by-rank. These improvements established the amortized time complexity $O(\alpha(n))$, where $\alpha(n)$ is the inverse Ackermann function. Since $\alpha(n) \leq 5$ for any practical dataset, numerous subsequent studies [32–34] have shown that Union–Find effectively operates in near-constant time, making it a fundamental solution to the Dynamic Connectivity Problem.

Union–Find consists of two core operations:

- **Find(x)**: Returns the representative (root or parent) of the set containing element x, enabling the system to determine whether two elements belong to the same group.
- **Union(x, y)**: Merges the sets containing x and y, ensuring both share a common representative.

To achieve high performance, two structural optimizations are commonly applied:

Path Compression During `Find`, all traversed nodes are directly linked to the root, flattening the tree and reducing future lookup depths.

Union by Rank When merging two sets, the tree with smaller rank (height or size) is attached beneath the tree with larger rank, preventing the formation of deep trees.

With both optimizations, the total time for performing m union and find operations is:

$$O(m\,\alpha(n))\,.$$

Given the extremely slow growth of $\alpha(n)$, Union–Find provides near-constant efficiency and is widely used in graph algorithms, social-network analysis, and large-scale clustering applications [34, 35]. Its suitability is particularly evident in scenarios requiring frequent merge and connectivity queries.

Union–Find in UF-FAE

In this study, Union–Find serves as the core grouping mechanism of the UF-FAE framework. By continuously monitoring sender–receiver relationships in transaction records, the system incrementally merges related accounts into evolving transaction groups. When a new transaction edge causes two previously separate groups to merge, the event is flagged as a potentially suspicious structural change, triggering subsequent feature extraction and anomaly detection.

UF-FAE first constructs weakly connected components (WCCs) using path-compressed Union–Find, after which the components are restored to directed form for graph-theoretic analysis. Node-level and group-level indicators are computed, while categorical and numerical fields from the raw dataset undergo feature encoding. This ensures that all graph-derived and attribute-based features are incorporated into downstream machine-learning models.

4.2. Topological Proof of the Equivalence Between Disjoint Sets and Union–Find

To explain and justify the relationship between disjoint-set partitions and the Union–Find (disjoint-set union, DSU) data structure, we show that the DSU algorithm maintains equivalence relations consistent with topological principles. Specifically, Union–Find dynamically preserves the path-connected components π_0 of a 1-dimensional simplicial complex. This subsection defines the equivalence relation, constructs the quotient space, and proves that the set partitions maintained by Disjoint Sets and by Union–Find are mathematically identical.

Objective: Prove the equivalence

$$\text{Disjoint Sets} = \text{Union–Find} = \pi_0(\text{path-connected components}).$$

Perspective: Computer Science + Topology.

Core Concept: Union–Find maintains π_0 of a 1-dimensional simplicial complex under dynamic connectivity.

Definition 1: Parent Map and Forest

Let $X = \{x_1, x_2, \ldots, x_n\}$.

Define the parent map:

$$p(x) = \begin{cases} x, & \text{if } x \text{ is a root,} \\ \text{parent}(x), & \text{otherwise.} \end{cases}$$

Define the edge set:

$$E = \{(x, p(x)) \mid x \in X,\ p(x) \neq x\}.$$

Then the forest structure is:

$$G = (X, E),$$

where each tree corresponds to one disjoint set.

Using a Python implementation:

```
def find(x):
    if parent[x] != x:
        parent[x] = find(parent[x])   # corresponds to p(parent(x))
    return parent[x]
```

Definition 2: Topological Modeling

Construct a 1-dimensional simplicial complex K from forest G.

Define the path-connected relation:

$$x \sim y \iff \exists (x = v_0, v_1, \dots, v_k = y) \text{ such that } (v_i, v_{i+1}) \in E(K),\ \forall i.$$

The path-connected components are:

$$\pi_0(K) = X/\sim .$$

Because $\sim$ is reflexive, symmetric, and transitive, it forms an equivalence relation. The quotient $X/\sim$ corresponds precisely to the set of connected components of K.

Path compression makes $p(x)$ converge toward a constant map (mapping almost all nodes directly to the root). In topological terms:

$$\text{Path Compression} = \text{Homotopy Collapse}$$

which collapses the space to its π_0 equivalence classes while preserving connectivity structure. This yields amortized time complexity $O(\alpha(m, n))$.

Definition 3: Find and Union Operations

$$\text{Find}(x) = \begin{cases} x, & p(x) = x, \\ \text{Find}(p(x)), & \text{otherwise.} \end{cases}$$

$$\text{Union}(x, y): \quad r_x = \text{Find}(x), \quad r_y = \text{Find}(y).$$

If $r_x \neq r_y$, attach the tree of smaller rank under the larger.

Path compression:

$$p(x) \leftarrow \text{Find}(p(x)).$$

Lemma 1: Find and Path Connectivity

$$\text{Find}(x) = \text{Find}(y) \quad \iff \quad x \sim y.$$

Proof.

($\Rightarrow$) If two nodes share the same root, there exists a path between them; hence they are connected.

($\Leftarrow$) If $x \sim y$, the merge operations forming this path place them in the same tree, hence the same root.

□

Lemma 2: Path Compression Preserves π_0

$$\pi_0(K') = \pi_0(K).$$

Reason: Collapsing edges within a tree does not change connectivity since trees are contractible spaces.

□

Lemma 3: Topological Meaning of Union

If two root nodes r_1 and r_2 are connected by a new edge:

$$\pi_0(K_{\text{after}}) = \pi_0(K_{\text{before}}) - 1.$$

Reason: Union creates a new edge between two components, merging them into a single component; the remaining components remain unaffected.

□

Theorem: Topological Equivalence

For any time t:

$$P_{\text{UF}}(t) \cong \pi_0(K_t).$$

Define the bijection:

$$\Phi_t : S_i \mapsto C_i,$$

where S_i is the i-th set in Union–Find and C_i is the i-th connected component of K_t.

Using Lemmas 1–3:

$$\text{Union–Find partition} = \text{Path-connected partition}.$$

Thus:

$$\text{UF} = \pi_0.$$

Corollary: Dynamic Connectivity Interpretation

$$\begin{aligned} \text{Find} &\rightarrow \text{query the component of a node}, \\ \text{Union} &\rightarrow \text{merge two components}, \\ \text{Path Compression} &\rightarrow \text{collapse edges within a tree}. \end{aligned}$$

Hence:

$$\boxed{\text{Union–Find dynamically maintains } \pi_0(K)}.$$

Conclusion: Three-Layer DAG Structure From Topological Equivalence

If the set partitions maintained by Disjoint Sets and Union–Find are mathematically identical:

$$P_{\text{Disjoint}} = P_{\text{Union–Find}},$$

then both structures produce identical evolution under Find and Union. Their complexity therefore agrees:

$$T_{\text{Disjoint}}(n) = T_{\text{Union–Find}}(n) = O(\alpha(n)).$$

Furthermore:

$$\text{Union–Find partition} = \pi_0(K_t).$$

Thus Union–Find is not merely a data structure; it is a discrete realization of π_0.

This equivalence induces three corresponding DAGs:

1. **Raw causal DAG** Formed by temporal transaction edges (sender → receiver). Represents the topological space K_0.
2. **Rank-filtration DAG** G_{UF} Each Union operation reduces π_0 by one:

   $$K_0 \subset K_1 \subset \cdots \subset K_t,$$

   and edges follow rank order:

   $$(u, v) \in E_{\mathrm{rank}} \iff \mathrm{Union}(u, v) \text{ with } r(u) \geq r(v).$$

   Since rank is monotonic, G_{UF} is a DAG.
3. **Homotopy-quotient DAG** G' Strongly connected components are collapsed:

   $$K' = K/\simeq,$$

   with edge set:

   $$E' = \{(C_i, C_j) \mid \exists (u, v) \in E,\ u \in C_i,\ v \in C_j\}.$$

   If $H_1(G') = 0$, then G' is a Directed Acyclic Quotient up to Homotopy (DAQH).

These correspond to the three manifestations of π_0:

Layer	DAG Representation	Topological Meaning
Static	Raw DAG	K_0 : real transaction flow
Evolutionary	Rank-filtration K_t	structure generation process
Homotopy	DAQH	$K' = K/\simeq$: stable inter-group topology

Python Illustration

```
def find(x):
    if parent[x] != x:
        parent[x] = find(parent[x])
    return parent[x]

def union(x, y):
    root_x = find(x)
    root_y = find(y)
    if root_x != root_y:   # anti-cycle guarantee
        parent[root_x] = root_y
```

The condition `if root_x != root_y` guarantees acyclicity:

- If two nodes already belong to the same set, `find()` returns the same root, preventing merging.
- If they belong to different sets, the union occurs only between the two roots, preventing cycle formation.

In the Union–Find data structure, certain intermediate operation sequences may temporarily produce trees whose height exceeds $O(\log n)$. However, this does not affect the amortized efficiency because each `find(x)` operation applies *path compression*, which collapses the search path to the root. Specifically, during `find(x)`, the algorithm recursively obtains the root of x and rewrites all traversed parent pointers:

$$\mathrm{parent}[v] = \mathrm{root}(v),$$

ensuring that the amortized cost remains bounded by $O(\alpha(m, n))$, where α is the inverse Ackermann function.

Furthermore, the forest structure maintained by Union–Find can never produce a cycle. This follows from three structural properties:

1. Unique-path property Each node has exactly one parent, and only root nodes are self-referential. Thus, every node has a unique upward path, preventing multiple routes that may form a cycle.

2. Disjointness of unions A `union(x, y)` operation only links the roots of two *different* sets. Since the new edge always connects two previously disjoint trees, it cannot introduce a backward reference that would create a closed loop.

3. Path compression shortens but never adds edges The `find()` operation merely replaces an existing parent pointer with one pointing directly to the root. This is equivalent to performing an edge contraction in topological terms. Because no new edges are introduced, and all modifications preserve the original direction toward the root, cycles cannot emerge.

Together, these properties ensure that a Union–Find forest remains acyclic under all valid sequences of operations, while path compression maintains its near-constant amortized performance.

4.3. Construction of One-Dimensional Eigenvectors

All input features are concatenated into a single one-dimensional vector:

$$X = [x_1, x_2, \ldots, x_n],$$

where X is composed of two major parts:

$$X = \left[X_{\text{num}} \parallel X_{\text{cat}}^{(\text{OHE})} \right],$$

with:

- X_{num}: numerical features,
- $X_{\text{cat}}^{(\text{OHE})}$: categorical features after One-Hot encoding.

(A) Numerical Features

- Transaction amount: `Amount`
- Graph-based group statistics: `group_node_count`, `group_edge_count`, `group_bidirect_ratio`
- Sender/receiver centrality indicators: `sender_degree`, `receiver_degree`, `sender_closeness`, `receiver_closeness` `sender_betweenness`, `receiver_betweenness`

Note: The fields `Date` and `Time` are used only for chronological splitting to generate timestamps and are excluded from the model features to prevent temporal leakage.

(B) Categorical Features (after OHE)

- Currencies: `Payment_currency`, `Received_currency`
- Locations: `Sender_bank_location`, `Receiver_bank_location`
- Payment method: `Payment_type`

The label variable is defined as:

$$y \in \{0, 1\},$$

where $y = 1$ denotes a laundering transaction corresponding to the field `Is_laundering`.

4.4. Categorical Feature Encoding Process (Algorithm 1)

String Indexer Each categorical variable C is first mapped to an integer index. If the categorical domain contains k categories $c_1, c_2, \ldots, c_k$, each category is assigned a unique integer in the range 0 to $k - 1$. For example:

$$\text{UK} \mapsto 0, \quad \text{UAE} \mapsto 1, \quad \text{China} \mapsto 2.$$

This step converts string-valued categories into a numeric format suitable for the downstream encoding process.

One-Hot Encoder After index assignment, each integer index is transformed into a One-Hot vector of length k. For a categorical index i, the One-Hot vector is a standard basis vector in $\mathbb{R}^k$:

$$\mathrm{OHE}(c_i) = [0, \ldots, 0, 1, 0, \ldots, 0].$$

Thus, categories are mapped from a discrete index set to a k-dimensional binary space. This transformation ensures that all categorical vectors are mutually exclusive and linearly independent, with equal Euclidean distances between any pair of category vectors. Consequently, the model does not mistakenly interpret category indices as having ordinal significance.

Vector Assembling Finally, all numerical features X_{num} and the One-Hot encoded categorical vectors are concatenated into the unified input feature vector X. This vectorization step enables downstream machine learning models to jointly process continuous and categorical attributes, completing the feature construction pipeline.

Algorithm 1 Categorical Feature Encoding

Categorical variable $C = \{c_1, c_2, \ldots, c_k\}$ Numerical feature vector $\mathbf{x}$

StringIndexer:

Map each category c_i to an integer index i where $i \in \{0, 1, \ldots, k-1\}$.
Example: UK $\rightarrow$ 0, UAE $\rightarrow$ 1, China $\rightarrow$ 2.

OneHotEncoder:

Define a mapping $\phi : \{0, 1, \ldots, k-1\} \rightarrow \{0, 1\}^k$ such that

$$\phi(i) = [\,0, \ldots, 0, \underbrace{1}_{(i+1)\text{-th position}}, 0, \ldots, 0\,],$$

where the $(i+1)$-th component equals 1 and all others equal 0.
Example: if $\mathrm{UK}_{\mathrm{idx}} = 2$, then

$$\phi(2) = [0, 0, 1, 0, \ldots, 0].$$

VectorAssembler:

Concatenate all numerical features and One-Hot encoded vectors into a single feature vector:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n].$$

ReturnX

4.5. Analysis of Clustering and Node Roles in Transaction Networks

To further analyze latent abnormal behaviours within transaction networks, this study applies the Union–Find algorithm to perform weakly connected component (WCC) clustering over the complete transaction dataset. Each account is treated as a node, and each transaction as a directed edge. Based on the account-level relationships in the SAML-D dataset, the original WCC-derived undirected graph is subsequently transformed back into a directed structure, followed by efficient path-compressed Union–Find operations. This procedure yields a total of 15,592 subgraph groups, where each group represents a reachable transactional cluster.

For each group, we compute the in-degree and out-degree of all constituent nodes. Nodes with zero in-degree are labelled as *Senders*, while nodes with zero out-degree are labelled as *Receivers*. This classification reveals structural role differentiation within each component. Initial results show that most groups contain large numbers of both Senders and Receivers, indicating that funds may circulate or propagate through multiple layers within the group. Such patterns resemble the classical "split–mix–recombine" behaviour characteristic of money-laundering operations. These structural observations reveal small-world properties and heterogeneous node roles in AML transaction graphs,

providing semantic context for further graph indicators such as betweenness centrality and bidirectional edge ratio. They also enhance the interpretability and extensibility of the proposed multi-modal feature integration framework.

To investigate whether large-scale, high-risk subgroups exist within the transaction network, this study examines the size distribution of all Union–Find-generated clusters. Groups are ranked in descending order by number of nodes, and the user may query the top N groups (with ties included) for focused analysis.

After selecting the top-N largest groups, the system computes the number of laundering transactions associated with each one. This is achieved by mapping each transaction's source account to its corresponding group and counting the edges labelled with `Is_laundering = 1`. The analysis reveals whether each high-density group contains laundering activity and additionally counts how many of the largest groups contain no laundering transactions at all.

This analysis strategy efficiently identifies suspicious behaviour embedded in large-scale transaction clusters. It enables investigators to focus resources on high-risk subgraphs. Experimental results show that the top 3% largest groups (465 groups in total) account for 6,236 laundering transactions, covering 63.12% of all laundering cases in the dataset. These findings indicate that high-density subgraphs exhibit significantly elevated risk levels and should be prioritized for regulatory monitoring.

The proposed method is flexible and scalable, serving as a "first-stage risk hotspot filter" within multi-modal AML systems. It allows decision-makers to quickly pinpoint potential threat zones within massive datasets, which can then be passed to machine-learning or deep-learning models for finer-grained prediction. An illustrative example and Algorithm 2 are provided below.

Directed Graph Formulation

Let the transaction dataset be represented as a directed graph:

$$G = (V, E),$$

where each node $v \in V$ denotes an account, and each directed edge $(u, v) \in E$ represents a transaction from account u to account v. Using Union–Find, the nodes of G are partitioned into disjoint subsets (groups) based on connectivity.

Initially, each node is an independent singleton set:

$$\forall v \in V, \quad \text{parent}(v) = v.$$

For each transaction edge (u, v), we perform:

$$\text{union}(u, v),$$

which merges the sets of u and v. With path compression, each node's representative (root) is computed as:

$$\text{root}(v) = \text{find}(v).$$

Thus, for any $u, v \in V$:

$$\text{root}(u) = \text{root}(v) \quad \Longleftrightarrow \quad u \text{ and } v \text{ belong to the same transaction-connected component.}$$

Counting Laundering Transactions per Cluster

For each cluster $C \subseteq V$, define the laundering count:

$$L(C) = \sum_{(u,v)\in E,\, u,v\in C} \mathbf{1}_{\{\text{Is\_laundering}(u,v)=1\}},$$

which measures the number of laundering-labeled transactions occurring entirely within cluster C.

This formulation provides a mathematically precise basis for identifying high-risk clusters within large transaction networks.

Table 1
Characteristics of UF-FAE as a Hybrid Approximation Algorithm

Feature	Description
Hybridization	Combines two or more heuristic techniques to achieve both global exploration (diversification) and local refinement (intensification).
Heuristic-driven	Uses domain knowledge and problem-specific rules to guide the search toward high-quality feasible solutions quickly, rather than seeking an absolute optimum.
Approximation	Produces solutions that approximate the optimal result, with theoretical or empirical evidence demonstrating near-optimal performance.

Algorithm 2 UF-AML Clustering Algorithm

Transaction dataset D with edges (u, v) and laundering labels $I_{\text{laundering}}$ Cluster assignments and laundering statistics
Initialize: Each account is placed in its own cluster.

for all transaction edges $(u, v) \in D$ **do** Union the clusters of accounts u and v.

for all accounts **do** Find the root cluster representative.
Calculate laundering transaction counts for each cluster.

Cluster assignments and laundering statistics.

4.6. Role of Machine Learning in UF-AML

In this study, machine learning does not participate directly in the operation of the proposed algorithm. Instead, it serves as a validation mechanism to evaluate the effectiveness of graph-theoretic indicators in money-laundering detection. Through the UF-AML (Union–Find-based Anti-Money Laundering) clustering algorithm, weakly connected components (WCCs) are constructed from the transaction network. Within each component, multiple graph-based indicators are computed, including centrality measures, reciprocity, and fan-in/fan-out ratios.

These graph indicators are then combined with the original transaction attributes and fed into machine-learning models. The performance of models enhanced with graph-theoretic features is compared against models trained solely on raw transaction attributes. Experimental results show that incorporating graph-based features significantly improves classification precision and recall. This demonstrates the capability of the UF-AML framework to reveal suspicious transactional groups and enhance the overall effectiveness of AML detection.

4.7. UF-FAE Architecture and Algorithm Localization

The proposed UF-FAE algorithm belongs to the class of heuristic-based hybrid approximation algorithms [36, 37]. It integrates structural heuristics from Union–Find with statistical approximations from machine-learning models, achieving both interpretability and computational efficiency. The key characteristics of this class of algorithms are summarized in Table 1.

UF-FAE integrates the structural heuristics of the Union–Find data structure with the statistical approximation capabilities of machine-learning models. By maintaining disjoint-set structures over transaction nodes, the algorithm dynamically tracks group-level connectivity within the distributed ledger technology (DLT) network. Based on these evolving groups, UF-FAE extracts graph-level descriptors that serve as the initial heuristic solution for downstream classification models, which then approximate feature relationships and identify anomalies.

The core of UF-FAE is a Union–Find backbone that maintains group merging and lookup operations in amortized time $O(\alpha(n))$. On top of this, the machine learning component performs multi-dimensional feature approximation and classification inference. This hybrid design preserves topological structure, supports high-frequency transactional environments, and provides near-optimal efficiency for detecting laundering behaviours.

Consequently, UF-FAE can be viewed as a hybrid algorithmic framework that combines structural heuristics with data-driven approximation, enabling robust, scalable, and interpretable AML detection over large-scale transaction networks.

4.8. Distinguishing Between Hybrid Approximation and Multimodal Learning

Although the proposed UF-FAE framework incorporates both structural algorithms and statistical learning models, its theoretical positioning lies firmly within heuristic-based hybrid approximation rather than multimodal machine learning. These two paradigms differ substantially in their objectives, integration mechanisms, and theoretical foundations.

Hybrid approximation algorithms belong to the domain of algorithm design and optimization theory. Their primary goal is to obtain near-optimal solutions within acceptable computational complexity. Such methods strategically combine multiple heuristic techniques to balance global exploration (diversification) and local refinement (intensification), a design commonly adopted for solving NP-hard approximation and search problems [36-38].

Within UF-FAE, Union–Find plays the role of a structural heuristic: through path compression, it maintains dynamic connectivity within the transaction network. The machine learning module functions as a statistical mechanism that performs classification and inference within the feature space. The overall design aims to achieve structural interpretability and near-optimal performance under low amortized time complexity, consistent with the principles of hybrid approximation.

In contrast, multimodal machine learning belongs to the fields of representation learning and data fusion. Its focus lies in integrating heterogeneous data modalities to learn unified cross-modal semantic representations [39]. Although this study incorporates both graph-theoretic features and transactional attributes into the model—forming what may be viewed as multimodal feature fusion—this integration occurs only at the data level. It does not involve cross-modal semantic alignment or shared latent-space learning, which are defining characteristics of true multimodal systems.

Therefore, UF-FAE should be understood as a hybrid approximation framework that combines structural heuristics with data-driven statistical approximation. The multimodal features serve solely as input-level augmentation to strengthen the classifier, rather than defining the underlying learning philosophy or model architecture.

5. Research Framework

5.1. Dataset

This study employs the Synthetic Transaction Monitoring Dataset for AML (SAML-D), published on the Kaggle platform. The dataset is a realistically constructed simulation of financial transactions, designed to emulate the statistical characteristics of genuine banking activity. Its structure spans multiple banks, customer profiles, and transaction channels, including high-frequency micro-transfers, cross-border transactions, and obfuscation of funding sources. These properties make it well-suited for high-dimensional, dynamic, and cross-domain transaction graph modeling, which aligns with the requirements of this research.

SAML-D contains 12 features and 28 transaction types, comprising 11 benign categories and 17 suspicious categories. These attributes were curated through existing public datasets, academic literature, and interviews with AML professionals. The dataset consists of 9,504,852 transactions, among which 0.1039% (9,873 transactions) are labeled as laundering activities[40]. The field `Is_laundering` serves as the ground-truth label for supervised learning and indicates whether a transaction is associated with money laundering behaviour.

For the modeling phase, this study designs three parallel experimental paths:

1. Machine learning models trained solely on graph-theoretic indicators.
2. Models trained solely on raw transactional attributes from SAML-D.
3. Multimodal feature fusion models that combine graph indicators with original transaction fields.

These configurations allow us to compare the contributions of different feature types to money-laundering detection. The evaluation metrics include AUC, precision, recall, and F1-score, forming the basis for a comprehensive ablation analysis.

The core algorithm of this work, UF-FAE (Union–Find-based Feature-Augmented Embedding), incorporates path compression and union-by-rank to partition large-scale transaction networks into weakly connected components, from which graph-theoretic and statistical group-level features are extracted. To assess the impact of these components, an ablation study is conducted by incrementally removing specific features or modules and measuring the resulting changes in predictive performance.

In addition, this study employs statistical analysis, unsupervised learning, and supervised learning to examine distributional patterns and structural relationships within the transaction dataset. Several baseline classification models

are included, such as logistic regression, decision trees, random forests, and linear support vector machines. Cross-validation is used to evaluate their predictive accuracy. This chapter presents the UF-FAE ablation results and the comparative performance of all machine learning models, thereby validating the effectiveness and feasibility of the proposed framework.

Overall, the research pipeline resolves common challenges in blockchain-oriented ETL processes, including data heterogeneity and loss of interaction structure, while maintaining scalability and analytical precision. The automated integration of graph-structural features provides more accurate representations of transaction paths and node roles. This, in turn, enhances downstream tasks such as anomaly detection and community discovery, enabling a broader set of blockchain analytics applications. The results are summarized as follows.

Algorithm 3 UF-FAE: Union–Find based Feature-Augmented Embedding

Transaction records $T = \{(s_i, r_i, a_i, t_i)\}$, where s_i and r_i are sender and receiver, a_i is amount, and t_i is timestamp
Feature-augmented embedding matrix $X \in \mathbb{R}^{n \times d}$, where n is the number of groups
Initialize: Create disjoint sets for all accounts using Union–Find.

for all transactions $(s_i, r_i, a_i, t_i) \in T$ **do** Perform Union operation: Union(s_i, r_i);
Group all transactions according to connected components (clusters).

for all groups G_j **do Extract structural features:**
- Number of nodes / edges
- Degree statistics (avg/max/min)
- Betweenness or closeness centrality (optional)

Extract transactional features:
- Mean / standard deviation of transaction amounts
- Temporal features (e.g., time-gap statistics)
- Entropy of sender / receiver diversity

Concatenate all features into a vector x_j;
Stack all vectors x_j into the final feature matrix X.

(Optional) Normalize or embed X for downstream machine-learning tasks.

ReturnX

5.2. Machine Learning Model Design for Multimodal Feature Fusion

This study further develops a multimodal machine-learning framework that integrates graph-theoretic features (e.g., number of nodes, number of edges, centrality metrics) with raw transactional attributes (e.g., amount, payment type, currency, geographical location). A strict time-series split strategy is adopted to emulate real-world AML monitoring environments, ensuring that models are trained only on past data while preventing future-information leakage. This design enhances the deployment credibility of the proposed detection system.

Feature processing and pipeline construction are implemented using PySpark. The `Date` and `Time` fields are first merged into a unified `timestamp` value and converted to an integer format for chronological sorting. The dataset is then ordered in ascending temporal sequence, with the earliest 80% used as the training set and the remaining 20% as the test set, thereby enforcing an irreversible temporal separation between training and evaluation.

For feature engineering, categorical variables (such as currency, location, and payment type) are encoded using `StringIndexer`, followed by expansion via `OneHotEncoder`. Numerical features are directly incorporated. All processed variables are concatenated into a unified `features` vector for downstream learning tasks.

Four widely used classification models are selected for comparative analysis:

- Logistic Regression,

- Decision Tree,
- Random Forest,
- Linear Support Vector Machine.

Each model is trained using the same feature set and the identical time-series data split. Model performance is evaluated on four key metrics: AUC, Precision, Recall, and F1-score. For classifiers capable of estimating feature importance (such as Decision Tree and Random Forest), the top ten most influential features are additionally extracted and reported.

6. Experimental Design and Results

This chapter describes the experimental setup, feature construction, ablation configuration, and comparative results. All evaluations use strict time-ordered splits to prevent future-information leakage, in accordance with standard AML monitoring requirements.

6.1. UF-FAE Ablation Experiment Analysis Results

This study adopts multiple supervised machine-learning models to analyze the distributional characteristics and structural dependencies of transactional data. These analyses provide the foundation for subsequent model construction and optimization. Cross-validation is used to evaluate classification performance. This section presents the ablation study results of the UF-FAE framework and compares the effectiveness of various machine-learning classifiers, thereby validating the robustness and feasibility of the proposed approach.

The overall research pipeline effectively resolves common challenges in blockchain-oriented ETL (Extract–Transform–Load), such as data heterogeneity and the loss of interaction structure, while maintaining both scalability and analytical precision. Through the automated integration of graph-structural features, the system captures monetary flow trajectories and node roles with higher fidelity, providing strong analytical foundations for downstream tasks including anomaly detection and community discovery.

The following table summarizes the results obtained from four machine-learning models across three feature configurations: multimodal fusion, raw transactional attributes, and graph-only features. All experiments use an 80/20 temporal data split, ensuring chronological consistency.

6.2. Benchmark Model Evaluation

This study evaluates four machine-learning classifiers across three feature configurations (raw features, graph-based features, and multimodal fusion) to examine their effectiveness in identifying Class 1 transactions (money laundering). The ablation experiments reveal the following findings.

Multimodal Features (Raw + Graph-Based)

- **Decision Tree** achieves the best performance under the multimodal setting, with Recall = 0.8637 and F1-score = 0.9182, substantially outperforming all other models. This demonstrates its ability to effectively capture suspicious laundering patterns.
- **Logistic Regression** performs well in overall ranking (AUC = 0.9866), yet its Recall for Class 1 is only 0.5558, indicating difficulty in identifying minority fraud cases.
- **Random Forest** and **Linear SVM**, despite achieving nearly perfect accuracy on Class 0, suffer from extremely low Recall on Class 1 (RF = 0.0070; SVM = 0.0019), rendering them ineffective due to severe overfitting to the majority class.

In terms of computational efficiency, Logistic Regression and Decision Tree offer the best balance between speed and performance.

Table 2
Performance comparison of UF-FAE ablation experiments across different feature sets.

Model	Group	AUC	C1 Precision	C1 Recall	C1 F1	W. Precision	W. Recall	W. F1	Top Features
Logistic Regression	Multimodal	0.9866	0.7467	0.5558	0.6372	0.9992	0.9993	0.9992	receiver_closeness, receiver_betweenness, sender_betweenness
Decision Tree	Multimodal	0.8621	0.9799	0.8637	0.9182	0.9998	0.9998	0.9998	sender_degree, sender_betweenness, receiver_degree
Random Forest	Multimodal	0.9913	1.0000	0.0070	0.0139	0.9989	0.9989	0.9983	receiver_degree, sender_degree, sender_betweenness
Linear SVM	Multimodal	0.9320	1.0000	0.0019	0.0037	0.9989	0.9989	0.9983	receiver_closeness, receiver_betweenness, sender_betweenness
Logistic Regression	Raw	0.7281	1.0000	0.0042	0.0084	0.9989	0.9989	0.9983	(No dominant features)
Decision Tree	Raw	0.4461	1.0000	0.0247	0.0483	0.9989	0.9989	0.9984	Amount, Payment_type_vec_6
Random Forest	Raw	0.7364	0.0000	0.0000	0.0000	0.9977	0.9989	0.9983	Amount, Payment_type_vec_6, Payment_type_vec_4
Linear SVM	Raw	0.6890	0.0000	0.0000	0.0000	0.9977	0.9989	0.9983	(No dominant features)
Logistic Regression	Graph-only	0.9786	0.9373	0.5166	0.6661	0.9994	0.9994	0.9993	receiver_closeness, receiver_betweenness, sender_betweenness
Decision Tree	Graph-only	0.9913	0.9803	0.8838	0.9296	0.9998	0.9998	0.9998	sender_degree, sender_betweenness, receiver_degree
Random Forest	Graph-only	0.9984	1.0000	0.9127	0.9544	0.9999	0.9999	0.9999	sender_degree, sender_betweenness, receiver_degree
Linear SVM	Graph-only	0.8980	0.0000	0.0000	0.0000	0.9977	0.9989	0.9983	receiver_closeness, receiver_betweenness, sender_betweenness

Table 3
End-to-end runtime (training + inference) of classical ML models across UF-FAE ablation groups.

Model	Multimodal	Raw	Graph-only
Logistic Regression	246.77 s	66.65 s	91.14 s
Decision Tree	227.93 s	91.37 s	65.22 s
Random Forest	883.70 s	675.01 s	571.89 s
Linear SVM (LinearSVC)	272.35 s	208.29 s	122.04 s

Note. Runtime reflects the total wall-clock time for **model training + prediction** on the test set within the Spark pipeline (including indexing, one-hot encoding, vector assembly, and model execution).

Raw Features Only

All models perform poorly when trained solely on raw transactional attributes. Class 1 Recall remains below 0.03 across all classifiers, with F1-scores approaching zero. Although the weighted metrics appear high, they are dominated by Class 0 predictions and therefore misleading. Feature analysis reveals that most models depend on a small number of categorical variables, and the feature `Amount` contributes negligibly, indicating that raw attributes alone lack sufficient discriminatory power.

Graph-Based Features Only

- Graph-based features yield overwhelmingly superior Class 1 detection performance compared to the other configurations.

- **Random Forest** achieves the strongest results, with Recall = 0.9127 and F1-score = 0.9544, showing that graph structures effectively expose latent laundering groups.
- **Decision Tree** remains consistently strong, with Recall = 0.8838 and stable performance across metrics, positioning it as a recommended model for practical AML deployment.
- **Linear SVM**, even with graph features, fails to identify Class 1 transactions, further confirming its limitations in modeling nonlinear anomalous patterns.
- **Logistic Regression** maintains moderate performance, offering interpretability and stability, making it suitable as a supplementary risk-control tool.

6.3. Discussion and Theoretical Implications

The proposed UF-FAE (Union–Find-based Feature-Augmented Embedding) framework contributes both theoretically and practically to AML detection. This work formalizes the social-network analysis pipeline required in AML as a family of algorithms with provably minimal time complexity. Traditional graph algorithms such as BFS, Dijkstra, betweenness, or closeness centrality are widely used in AML research, but their computational cost becomes prohibitive in high-frequency, cross-bank, microtransaction environments.

A key insight of this study is that core AML detection tasks fundamentally revolve around *connectivity queries* and *group merging events*, which can be solved by the Union–Find data structure in amortized $O(\alpha(n))$ time—the smallest known complexity for dynamic connectivity problems. UF-FAE leverages Union–Find not merely as an engineering construct, but as a topologically grounded mechanism.

To ensure mathematical rigor, this study introduces a topological proof showing that the disjoint-set structure maintained by Union–Find is equivalent to the topological invariant π_0 (path-connected components). Thus, Union–Find represents not only a data structure but a discrete realization of fundamental topological connectedness. This implies that each group-merging event in AML (due to a new transaction linking previously separate clusters) corresponds to an update of path-connected components, efficiently captured in near-constant time. This theoretical foundation explains UF-FAE's real-time scalability across large, dynamic, multi-bank transaction networks.

On the implementation side, temporal splitting, ablation experiments, and multi-model comparisons demonstrate that UF-FAE's group-level and graph-based features significantly improve machine-learning performance, especially in terms of Recall and AUC. Moreover, group-merging events themselves serve as strong signals of suspicious behaviour, and Union–Find provides the optimal tool for capturing such events.

6.4. Summary of Contributions

UF-FAE provides three primary contributions:

1. **Minimal time complexity for AML social-network analysis:** This study establishes the first algorithmic family for AML that achieves near-constant time complexity $O(\alpha(n))$, enabling scalable, real-time detection without relying on expensive full-graph computations.
2. **Topological formalization of AML group evolution:** Through rigorous proofs, Union–Find is shown to be mathematically equivalent to the topological invariant π_0, offering a unified and theoretically grounded framework for modeling AML connectivity events.
3. **Empirical validation on real-world-scale data:** Experiments confirm that integrating lightweight connectivity analysis with machine learning dramatically improves recall and robustness in AML detection, demonstrating that UF-FAE is both practical and theoretically sound.

Overall, UF-FAE suggests a new design philosophy for next-generation AML systems: rather than relying on expensive global graph algorithms, effective AML detection can be achieved through *topological connectivity*, *dynamic group features*, and *lightweight classifiers*. This approach offers interpretability, scalability, and deployability for banks and regulators in large-scale transactional environments.

References

[1] A. Abhinaya, "Influence of the COVID-19 Pandemic on Financial Crimes," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 6, no. 6, pp. 3427–3439, 2024. Available: https://www.irjmets.com/uploadedfiles/paper/issue_6_june_2024/59496/final/fin_irjmets1719342103.pdf.

[2] B. Deprez, B. Baesens, T. Verdonck, and W. Verbeke, *GARG-AML against Smurfing: A Scalable and Interpretable Graph-Based Framework for Anti-Money Laundering*, 2025.

[3] Feedzai, "The Future of AML: New Insights from Feedzai's 2023 Report." Available: https://www.feedzai.com/blog/the-future-of-aml-new-insights-from-feedzais-2023-report.

[4] Z. Tian *et al.*, "Towards Collaborative Anti-Money Laundering Among Financial Institutions," in *Proceedings of the ACM Web Conference 2025*, 2025.

[5] United Nations Office on Drugs and Crime, "Money laundering," 2025. Available: https://www.unodc.org/unodc/en/money-laundering/overview.html.

[6] M. Starnini *et al.*, "Smurf-Based Anti-money Laundering in Time-Evolving Transaction Networks," in *Machine Learning and Knowledge Discovery in Databases*, Springer, 2021, pp. 171–186.

[7] E. Altman *et al.*, "Realistic synthetic financial transactions for anti-money laundering models," in *NeurIPS*, New Orleans, USA, 2023.

[8] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[9] C.-C. Huang and A. Trangle, "Anti-Money Laundering and Blockchain Technology," *AML Case Study*, 2020.

[10] S. Aidoo, "The Role of Blockchain in AML Compliance: Potential Applications and Limitations," 2025.

[11] A. Oad *et al.*, "Blockchain-Enabled Transaction Scanning Method for Money Laundering Detection," *Electronics*, vol. 10, no. 15, 2021.

[12] Lucinity, "Data Silos and AML Compliance," 2024. Available: https://www.lucinity.com/blog/data-silos-and-aml-compliance.

[13] A. H. Pareja *et al.*, "EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs," in *AAAI*, 2020, pp. 5363–5370.

[14] Neo4j, "Financial Fraud Detection with Graph Data Science," Whitepaper, 2021. Available: https://neo4j.com/wp-content/uploads/2021/03/Financial-Fraud-Detection-with-Graph-Data-Science-Whitepaper.pdf.

[15] Neo4j, "Accelerate Fraud Detection With Graph Databases," 2023. Available: https://neo4j.com.

[16] K. Song *et al.*, "Identifying Money Laundering Subgraphs on the Blockchain," in *ACM AI in Finance*, 2024.

[17] V. Van Vlasselaer *et al.*, "APATE: A novel approach for automated credit card transaction fraud detection using network-based extensions," *Decision Support Systems*, vol. 75, pp. 38–48, 2015.

[18] M. Weber *et al.*, "Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks," in *KDD Workshop on Anomaly Detection in Finance*, 2019.

[19] M. Jullum *et al.*, "Detecting money laundering transactions with machine learning," *Journal of Money Laundering Control*, vol. 23, no. 1, pp. 173–186, 2020.

[20] L. Akoglu, H. Tong, and D. Koutra, "Graph-based Anomaly Detection and Description: A Survey," arXiv:1404.4679, 2014.

[21] X. Wang and G. Dong, "Research on Money Laundering Detection Based on Improved MST Clustering," in *KAM*, 2009, pp. 62–64.

[22] X. Li *et al.*, "Intelligent Anti-Money Laundering Solution Based on Novel Community Detection," in *CBD*, 2017, pp. 176–181.

[23] S. Huang *et al.*, "TMAS: A transaction misbehavior analysis scheme for blockchain," *Blockchain: Research and Applications*, vol. 5, no. 3, 2024.

[24] M. Hasan *et al.*, "Detecting anomalies in blockchain transactions using ML classifiers and explainability," *Blockchain: Research and Applications*, vol. 5, no. 3, 2024.

[25] I. Nti *et al.*, *Multi-Source Explainable AI for Blockchain Transaction Manipulation Detection*, 2025.

[26] S. Popov, "The Tangle: A Validity–Based Ledger Structure for IoT," IOTA Foundation, 2017. Available: https://cryptoverze.s3.us-east-2.amazonaws.com/wp-content/uploads/2018/11/10012054/IOTA-MIOTA-Whitepaper.pdf.

[27] N. Sealey, A. Aijaz, and B. Holden, *IOTA Tangle 2.0*, 2022.

[28] S. Brin and L. Page, "The anatomy of a large-scale Web search engine," *Computer Networks*, vol. 30, no. 1, pp. 107–117, 1998.

[29] R. E. Tarjan, "Efficiency of a good but not linear set union algorithm," *Journal of the ACM*, vol. 22, no. 2, pp. 215–225, 1975.

[30] M. Fredman and M. Saks, "The cell probe complexity of dynamic data structures," in *STOC*, 1989.

[31] B. Galler and M. Fisher, "An improved equivalence algorithm," *Communications of the ACM*, vol. 7, no. 5, pp. 301–303, 1964.

[32] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, 1979.

[33] R. Sedgewick and K. Wayne, *Algorithms*, 4th ed., Addison-Wesley, 2011.

[34] T. Cormen *et al.*, *Introduction to Algorithms*, 4th ed., MIT Press, 2022.

[35] J. Holm, K. Lichtenberg, and M. Thorup, "Poly-logarithmic deterministic fully-dynamic algorithms for connectivity," *Journal of the ACM*, vol. 48, no. 4, pp. 723–760, 2001.

[36] M. Lozano and C. García-Martínez, "Hybrid metaheuristics with evolutionary algorithms," *Computers & Operations Research*, vol. 37, no. 3, pp. 481–497, 2010.

[37] V. Máximo and M. Nascimento, "Intensification, learning and diversification in a hybrid metaheuristic," *Journal of Heuristics*, vol. 25, pp. 539–564, 2019.

[38] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.

[39] T. Baltrusaitis, C. Ahuja, and L. Morency, "Multimodal Machine Learning: A Survey and Taxonomy," *IEEE TPAMI*, vol. 41, no. 2, pp. 423–443, 2019.

[40] B. Oztas *et al.*, "Development of a Synthetic Transaction Monitoring Dataset," in *IEEE ICEBE*, 2023.

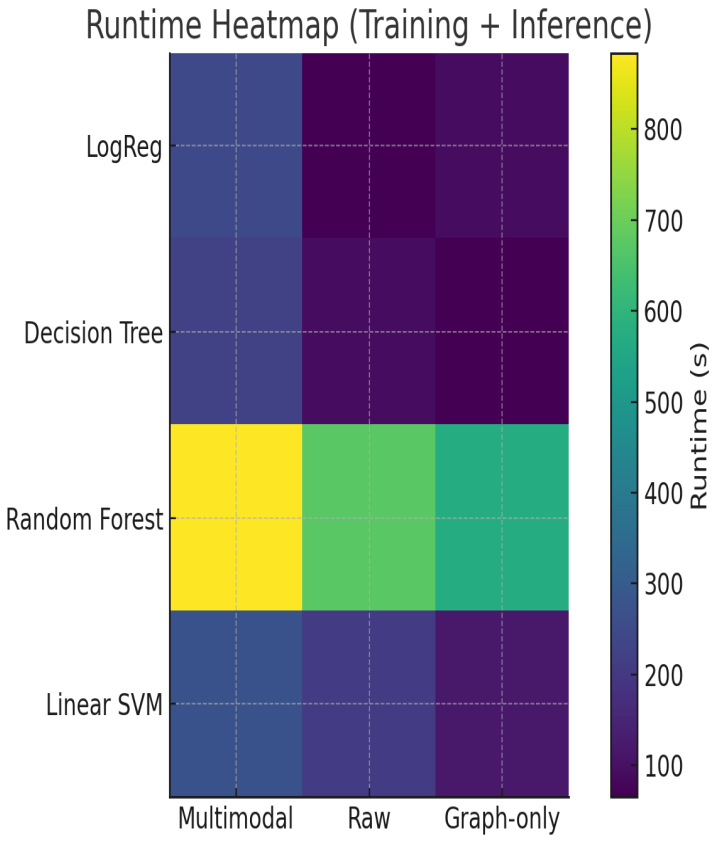

Figure 1: Runtime heatmap (training + inference) across UF-FAE ablation groups.