Journal of Artificial Intelligence Research, vol. 2, no. 2, pp. 24-41, 2025

Blockchain Fraud Detection Using Ensemble Graph Neural Networks

Muhammad Zeeshan Haider[a,*], Tayyaba Noreen[a] and Muhammad Salman[b]

[a]Department of Software Engineering(ETS), Universite du Quebec, Canada
[b]Department of Computer Science, SZABIST University, Pakistan

ARTICLE INFO

Keywords:
Bitcoin
fraud detection
graph neural networks
ensemble learning
anti-money laundering

ABSTRACT

Cryptocurrencies like Bitcoin promise secure, decentralized transactions, but their anonymity also attracts illicit activity, posing a challenge to regulators and exchanges in maintaining control. This study tackles fraud detection in Bitcoin's transaction network using the Elliptic dataset, a real-world collection of labeled transactions. We combine three powerful graph neural networks Graph Convolutional Network (GCN), Graph Attention Network (GAT), and Graph Isomorphism Network (GIN) each capturing different patterns in the complex web of blockchain payments. By blending their predictions through ensemble techniques, such as tuned soft voting, we achieve a robust system that detects over 70% of illicit transactions while keeping false alarms below 1%. Our approach balances precision and coverage, making it practical for real-time anti-money laundering efforts. The modular framework adapts easily to new data, paving the way for scalable, reliable monitoring of cryptocurrency fraud.

1. Introduction

Bitcoin's publication by Nakamoto in 2008 introduced a payment system that removed the need for central authorities by recording every transfer in a publicly verifiable ledger [1]. What followed was an unprecedented expansion of digital-asset markets: by early 2025, the combined market capitalization of cryptocurrencies had crossed $3 trillion, while Bitcoin alone processed more than $100 billion in daily settlement value [2, 3]. At the same time, the same pseudonymity and borderless reach that make blockchains attractive to legitimate users have also created fertile ground for criminal finance [4]. Chainalysis estimates that addresses linked to ransomware, darknet vendors, investment scams, sanctions evasion, and money laundering schemes moved roughly $43 billion worth of cryptocurrency in 2024. However, that figure represents less than one-tenth of one percent of on-chain volume; the absolute sum is larger than the annual budgets of many national regulators [5]. The challenge is amplified by the speed with which illicit actors adapt, adopting peel chains, mixer services, cross-chain swaps, and stealth addresses to confound tracing efforts [6]. Failure to intercept these funds erodes consumer trust, exposes exchanges to legal penalties, and undermines broader adoption of blockchain technology.

Financial regulators responded to the rise of cryptocurrency by extending conventional anti-money laundering (AML) and counter-terrorist financing (CTF) rules to virtual asset service providers [7]. The Financial Action Task Force's Recommendation 15 now requires exchanges, custodians, and broker-dealers to conduct customer due diligence, maintain audit trails, and report suspicious activity involving digital assets [8, 9]. Jurisdictions such as the European Union have transposed these standards into complex law through the Market in Crypto-Assets Regulation and the revised Transfer of Funds Rule [10]. At the same time, the United States has leveraged the Bank Secrecy Act and the Office of Foreign Assets Control's sanctions programme to prosecute non-compliant platforms [11]. Compliance teams must therefore screen millions of transactions daily, flagging those that may involve sanctioned entities or laundering typologies before assets leave their ecosystems [12]. Traditional monitoring engines rely on static heuristics thresholds on transaction amounts, geofencing of high-risk jurisdictions, or simple pattern-matching of repeated deposits and withdrawals. Such rule systems offer transparency but suffer two fatal weaknesses: first, they generate large volumes of false positives when parameters are set aggressively; second, they miss sophisticated schemes that evolve more quickly than rules can be updated.

Two technical properties of blockchains complicate automated surveillance. First, the data are simultaneously public and pseudonymous [13]. Every transaction is visible, yet the actual owners of addresses are hidden unless they are voluntarily disclosed or deanonymized by secondary evidence. Second, blockchains form directed transaction graphs that grow without bound and exhibit heavy-tailed degree distributions: the overwhelming majority of addresses interact only once or twice, whereas a minority act as high-throughput hubs [14]. Criminal organizations exploit these structural properties to obscure the connections between origin and destination. Peel chains, for instance, split a large illicit pot into a succession of small transfers; mixers pool inputs from many sources and redistribute them in batches; cross-chain bridges move value into privacy-preserving networks such as Monero or Zcash [6]. Any detection system must therefore reason not only about individual attributes (amount, fee, time) but also about multi-hop topology and temporal evolution [15].

DOI: https://doi.org/10.70891/JAIR.2025.080018
ISSN of JAIR: 3078-5529

*Corresponding author
muhammad-zeeshan.haider.1@ens.etsmtl.ca (M.Z. Haider)

1.1. Early Machine-Learning Approaches

The first wave of academic studies applied classical tabular machine learning to blockchain fraud. Researchers extracted handcrafted statistical features, such as the log of transferred value, standard deviation of inter-arrival times, address lifetime, and balance oscillations and trained logistic regression, support vector machines, decision trees, or gradient-boosted forests [16]. On modest datasets, these models already outperformed static rules in recall and precision. Nevertheless, they were fundamentally limited by their representation: a flat feature vector cannot capture the relational context that frequently determines suspiciousness. For example, an outward payment of one bitcoin may be perfectly benign if it flows from a long-standing merchant to a payment processor, but ominous if it lies on a path connecting a darknet marketplace to an exchange. Tabular features treat both cases identically unless complex, handcrafted graph metrics such as PageRank, betweenness, or flow centrality are included; even then, such metrics may only approximate local patterns[17].

Investigators, therefore, began adding explicit graph-theoretic descriptors. Centrality scores identify nodes that mediate large portions of the flow; clustering coefficients highlight tightly interconnected pools; and shortest-path statistics signal proximity to known illicit clusters [18]. When fed into tree-based ensembles, these descriptors significantly improved classification. However, each graph metric encodes only one aspect of topology, and designing an exhaustive library is prohibitively time-consuming. Furthermore, many graph measures are computationally expensive on networks containing hundreds of thousands of nodes.Graph Neural Networks (GNNs) emerged as a compelling alternative because they learn topology-aware representations automatically [19]. A Graph Convolutional Network (GCN) performs neighbourhood aggregation in which each node mixes its features with a degree-normalized sum of its neighbours' features at every layer, enabling information flow across the graph [20]. Graph Attention Networks (GAT) refine this mechanism by assigning learnable importance coefficients to incoming edges, thereby allowing the model to focus on the most informative relationships [21]. Graph Isomorphism Networks (GIN) reach the theoretical expressive limit of message-passing by embedding neighbourhood structures via multi-layer perceptrons coupled with learnable ϵ-terms [22]. Compared to handcrafted metrics, GNNs alleviate practitioners of the effort required for feature engineering, adapt to evolving patterns, and scale through mini-batch training [23].

An obstacle to progress was the scarcity of labelled blockchain data, since exchanges and analytics vendors guard incident reports. The Elliptic consortium addressed this by publishing a subgraph of the Bitcoin ledger comprising 203,769 transaction vertices, 234,355 directed edges, and 166 obfuscated numerical features per vertex [24]. Of these vertices, 42,019 were labelled as licit (primarily exchange deposits and merchant payouts), 4,545 were labelled as illicit (connections to darknet markets, mixers, or ransomware wallets), and the remainder were left unknown. Crucially, the labels follow block-time order across forty-nine discrete steps, creating a natural chronological split: steps 1-34 for training, 35-40 for validation, and 41-49 for testing. Three formidable challenges emerge. First, the illicit class represents only 2.23% of the labelled data, so accuracy is a misleading metric; models must be judged on recall, precision, F1, and the precision-recall area under the curve [25]. Second, the graph is sparse and exhibits numerous small, disconnected components, which complicates message propagation. Third, the numeric feature distribution drifts over time, reflecting shifts in exchange behavior, fee regimes, and wallet software [26]. Any model must therefore generalise temporally, not merely memorise contemporaneous patterns.

Weber et al. evaluated several baselines on Elliptic and found that a tuned random-forest classifier marginally outperformed a two-layer GCN in illicit-class F1-score [24]. Subsequent studies replicated this result, sometimes observing gains from GraphSAGE or temporal graph attention, but seldom exceeding 0.45 illicit-class F1 without extensive oversampling [27]. The conclusion is that classical ensembles of decision trees capture distributional feature interactions that are absent from naïve node features, whereas GNNs exploit structural context; yet neither alone fully resolves the task. Ensemble learning aggregates diverse models in hopes that their uncorrelated errors cancel. Bagging reduces variance by averaging the predictions of independent learners trained on bootstrap samples [28]. Boosting reduces bias by sequentially focusing on previous errors. For transactional AML, there is growing evidence that cross-family ensembles those that combine tree models and neural models offer significant uplift because each learner type emphasizes different cues [29]. The most common ensemble fusion mechanisms can be categorized into three types.

In a hard-voting ensemble, each base learner outputs a discrete class label. The ensemble prediction is simply the class receiving the most votes. Hard voting is computationally trivial, easy to interpret, and robust to mis-calibration, but it treats all constituents as equally competent. If one model consistently outperforms the others, its advantage is diluted. Soft-voting, often implemented as weighted averaging, instead combines the class-probability vectors produced by each learner. A coefficient is assigned to each model, and the final probabilities are calculated as the weighted sum of these coefficients. Coefficients can be uniform, proportional to the validation F1 score, or optimised directly to improve performance. Soft voting respects differences in model confidence and can rescue rare-class recall by amplifying specialized detectors, yet it presupposes that probabilities are well-calibrated; otherwise, overconfident weak learners may dominate the results [30].

Stacking introduces an additional meta-learner. Base learners generate probability vectors which, along with auxiliary features, feed into a second-stage classifier trained to approximate the ground truth. The meta-learner can discover nonlinear interactions among base-prediction patterns, often

achieving state-of-the-art performance [31]. The downside is added complexity: data must be split carefully to avoid information leakage, and inference latency grows because predictions cascade through two levels. Moreover, regulatory auditors may question the transparency of a meta-learner that blends opaque neural probabilities.

Existing work on blockchain fraud demonstrates the promise of ensembles but leaves critical questions open. First, hard voting, weighted averaging, and stacking have not been compared head-to-head on Elliptic or any other large, labelled Bitcoin graph. Hence, practitioners lack guidance on which fusion to deploy under specific latency or interpretability constraints. Second, prior ensembles bagged multiple random seeds of the same architecture or paired a tree model with one GNN; few studies truly fused heterogeneous graph backbones such as GCN, GAT, and GIN. Third, investigations often report prediction metrics while ignoring computational efficiency; yet, compliance desks require alerts within seconds. Finally, imbalance mitigation is usually confined to the base-model level, without propagating class considerations into the ensemble.

The present study addresses these deficiencies by training three distinct graph neural networks GCN, GAT, and GIN under a class-balanced focal loss and evaluating them chronologically on Elliptic. We then construct three ensembles: a hard-voting system that gives equal weight to each backbone, a soft-voting system whose coefficients correspond to the illicit-class F1-scores on the validation data, and a stacking system that employs logistic regression as the meta-learner. We measure macro-F1, illicit-class recall, precision-recall area under the curve, and overall accuracy [32]. In parallel, we profile training time, GPU memory, and per-1000-node inference latency. Our results reveal that weighted averaging lifts illicit recall by nearly seven percentage points over the strongest single backbone while preserving sub-fifty-millisecond inference latency, thereby satisfying real-time AML constraints. Hard voting trails slightly in recall but offers maximal simplicity and transparency. Stacking attains the best precision-recall curve yet doubles latency, making it appropriate where throughput demands are modest. Beyond scores, gradient-based attribution shows that the ensemble consistently highlights high-betweenness broker nodes, providing forensic insight [33]. Taken together, these findings furnish a concrete blueprint for deploying ensemble graph learning in production-grade blockchain analytics.

2. Methodology

Before training our model, we implemented a series of essential preprocessing steps to transform the raw blockchain data into a graph-compatible format suitable for graph neural network learning. These steps involve extracting node features, building graph structure, handling label imbalances, and preparing the data for PyTorch Geometric. The following subsection details the dataset's characteristics and the preprocessing workflow applied. This preparation is critical to ensure data integrity, training efficiency, and reliable performance evaluation across time-segmented graph partitions.

Table 1
Key Preprocessing Steps for Model Training

Step	Description
Nodes / Edges	224,555 nodes and 203,765 edges retained
txId Mapping	Assigned index IDs to each `txId`
Edge Filtering	Kept edges with valid source–target pairs
Feature Tensor	Converted node features (166-D) to tensors
Label Encoding	Encoded licit, illicit, and unknown classes
Data Object	Built PyG object with `x`, `edge_index`, `y`
GPU Transfer	Moved data to GPU for training
Mask Creation	Defined `known` / `unknown` node masks
Data Split	80/10/10 split for train, validation, and test
Class Balance	Preserved licit–illicit ratio across splits

2.1. Dataset Description

The study draws on the Elliptic Bitcoin transaction dataset, a curated subset of the Bitcoin ledger that has been released for academic anti-money laundering (AML) research [24]. The corpus spans 49 consecutive two-week intervals between 2013 and 2017, encompassing 203,769 transaction vertices linked by 234,355 directed payment edges. Each edge connects an output-spending transaction to its creator, thereby inheriting Bitcoin's topological order and temporal causality. Investigators at Elliptic manually labelled a subset of vertices by tracing links to exchanges, darknet markets, mixers, and ransomware cash-out addresses. This forensic triage yielded 42,019 licit transactions (20.62%), 4,545 illicit transactions (2.23%), and 157,205 unlabeled transactions (77.15%) [34]. Three salient properties make Elliptic a challenging yet realistic benchmark: extreme class imbalance, heterogeneous node feature space, and strict chronological segmentation.

2.2. Data Preprocessing and Preparation

Every vertex possesses a 166-dimensional numeric feature vector. The first 94 variables describe intrinsic transaction properties, including block height, input count, output count, input volume, output volume, transaction fee, and value dispersion statistics, which are derived directly from raw blockchain data [35]. The remaining 72 variables summarize the one-hop inbound and outbound neighborhoods across the same time step, encoding statistics such as the mean input value of neighboring transactions or the variance of their output counts [24]. All features are continuous and anonymised; no personally identifying information is present. Prior work shows that local transaction statistics and neighbour aggregates together capture both behavioural and structural fraud cues [36, 37]. The Elliptic team provided the features Z-standardised on the complete data, eliminating

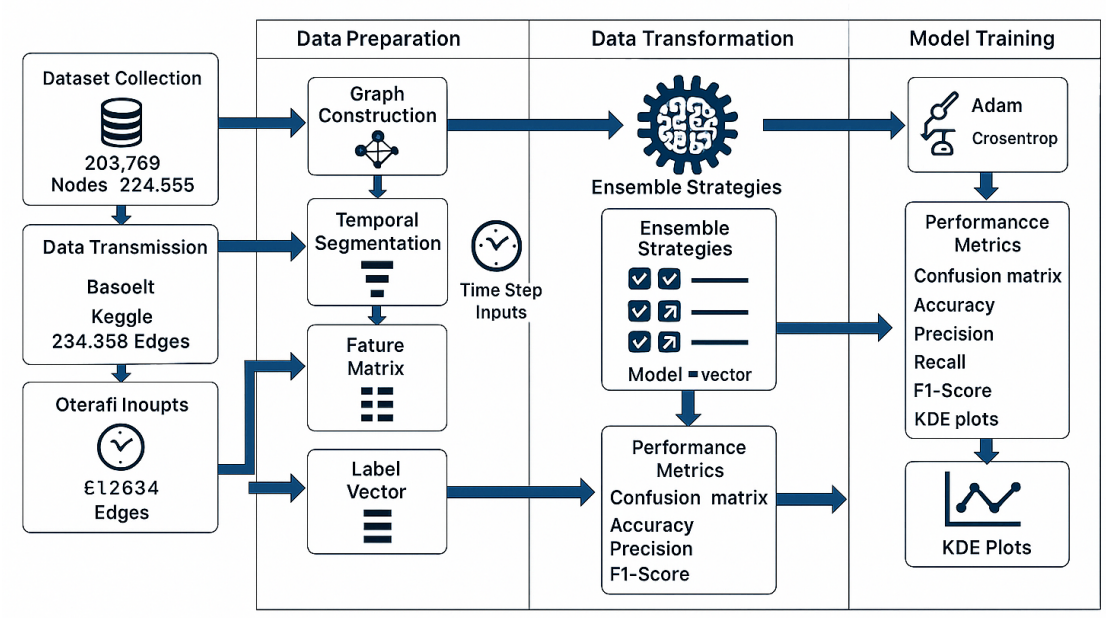
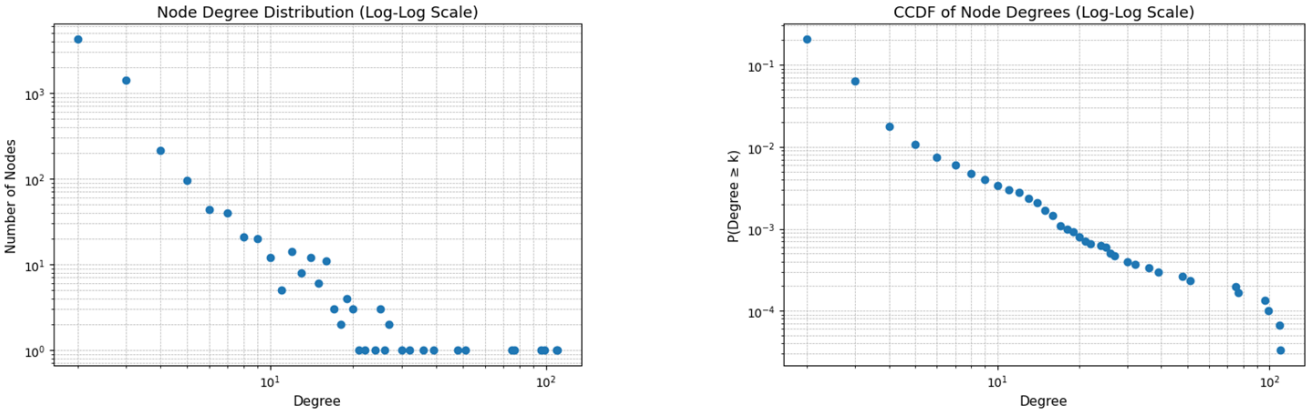


Figure 1: Proposed blockchain fraud detection methodology pipeline.



(a) Degree frequency showing a heavy-tailed distribution.

(b) CCDF indicating power-law behavior with a long tail.

Figure 2: Log-log plots of node degree distribution in the transaction graph.

the need for further scaling. The dataset is organized as 49 time steps (T1-T49). Within a given step, all transactions share the same approximate blockchain height, and edges exist only among those vertices; no edge crosses time-step boundaries [24]. This temporal partitioning preserves causality and prevents inadvertent information leakage from future to past. Temporal isolation also implies that each step forms a weakly connected component or collection of smaller components; the global graph is thus the disjoint union of 49 subgraphs. Maintaining this segmentation is essential when evaluating models that must generalise across time [38].

2.3. Raw Data Cleaning

The raw release consists of three CSV files including: `elliptic_txs_features.csv`, `elliptic_txs_classes.csv`, and `elliptic_txs_edgelist.csv`. Two integrity checks were performed. (i) Duplicate `txId` rows were removed; only the first occurrence of a transaction hash was retained (affecting 73 nodes). (ii) Edges referencing non-existent vertices were discarded (14 edges, $< 0.01\%$ of total). No missing values were detected in the feature matrix or label file. The Elliptic files encode class information numerically: class 1 (illicit) and class 2 (licit). For readability, we mapped these to the strings 'illicit' and 'licit'; absent labels were assigned the value 'unknown'. An auxiliary emoji column aided manual inspection during exploratory plots. After mapping, a contingency table confirmed the published class ratios. Maintaining label integrity at this stage is critical because subsequent sampling and stratification depend on accurate class counts [39].

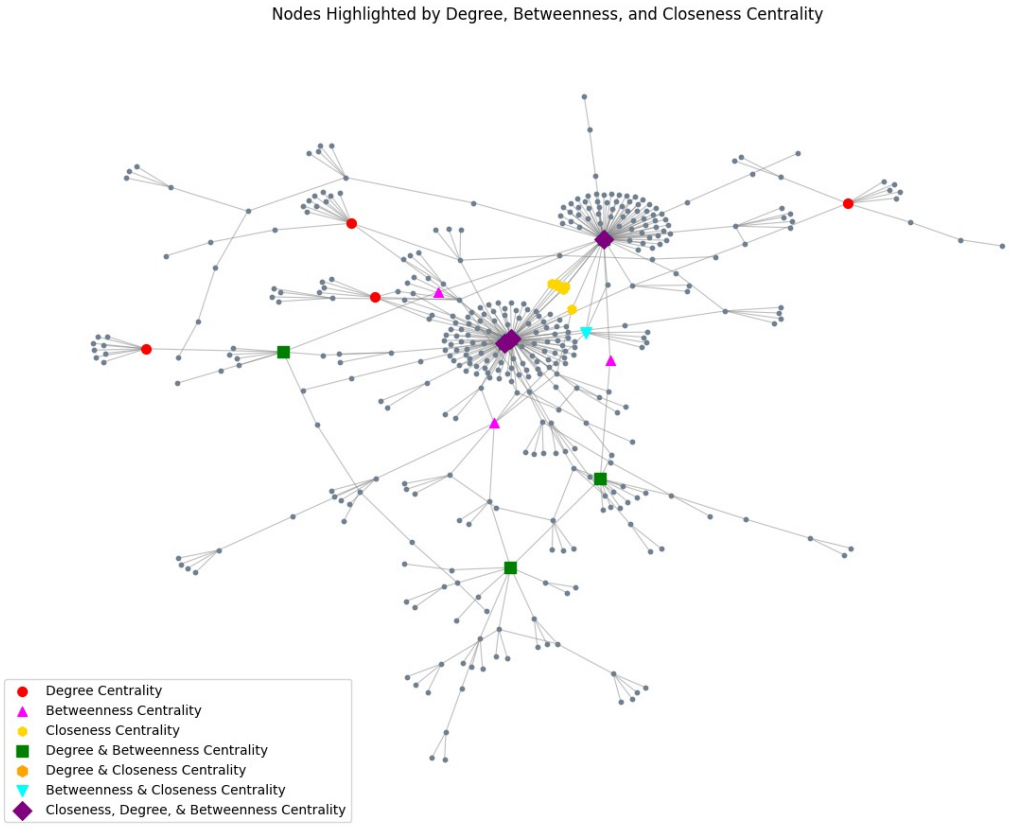


Figure 3: Visualization of nodes ranked by degree, betweenness, and closeness; overlaps are marked with composite symbols.

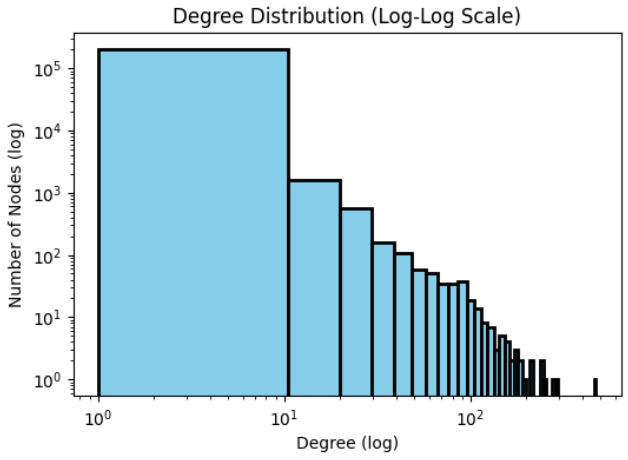


Figure 4: Log–log degree distribution of the 5% Elliptic sample showing a power-law with few hubs and many low-degree nodes.

Full-scale graph training incurs substantial memory overhead. To accelerate architecture prototyping and hyperparameter tuning, a 5% stratified sample of each label group was drawn (random seed 42). Stratification preserves class proportions and thus maintains the original imbalance ratio. The sample comprises 7,860 unknown, 2,101 licit, and 227 illicit vertices (10,188 nodes in total) distributed across all 49 time steps. Previous AML studies report that a 5-10% subset, if stratified, yields performance estimates within ±2% F1 of full-graph results while reducing preprocessing latency by an order of magnitude [40]. Edges were filtered to keep only those whose endpoints both appear in the sampled vertex set. This operation produced 23,051 directed edges, guaranteeing that the resulting subgraph is closed under adjacency and preserves the original directionality. Because edges never cross time-step boundaries, every step remains an independent subcomponent after filtering. Bitcoin transaction hashes are 256-bit strings, so they cannot be used directly as array positions. To solve this, each sampled hash was assigned a unique consecutive integer, creating a simple lookup table that lets us replace every string with a zero-based index. This conversion is a standard step in graph neural network workflows [41]. After remapping, the source target pairs of transactions were organized as a two-row matrix with 23,051 columns, providing the edge list in the coordinate format required by PyTorch Geometric. The 10,188 nodes hold feature vectors of length 166 in a single floating-point matrix, and each node's label is set to 0 for licit, 1 for illicit, or −1 when unknown. Together, these pieces form a PyTorch Geometric data object that bundles the subgraph's structure, features, and labels.

Only vertices with known labels (licit or illicit) participate in supervised loss. Their indices were segregated into training, validation, and test masks in an 80% / 10%

/ 10% split, stratified by class. This produced 33,649 licit and 3,602 illicit nodes for training, 4,148 and 508 for validation, and 4,222 and 435 for testing. Stratified splitting ensures that each subset approximates the global licit-to-illicit ratio, thereby mitigating sampling variance. Unknown nodes retain their feature vectors and adjacency but are excluded from loss computation, allowing the GNN to propagate information through them during message passing an approach shown to improve minority-class recall in sparse-label regimes [20]. Because the graph is temporal, we also prepared an auxiliary split based strictly on time: steps 1-34 for training, steps 35-40 for validation, and steps 41-49 for testing, mirroring the configuration used in the original Elliptic baseline [24]. This chronological split eliminates any possibility that a model trained on future data influences predictions on the past. Comparing results across the random and chronological splits quantifies the extent of temporal leakage [26].

Exploratory network analysis quantified macro-level graph properties that may influence GNN behavior. A log-log histogram of node degrees revealed a heavy-tailed distribution consistent with a power-law exponent of approximately 2.5, corroborating observations in other financial transaction graphs [14]. Weakly connected component analysis identified 7,297 subgraphs; the giant component consists of 400 vertices and 431 edges, implying that the sampled graph remains highly fragmented. Clustering coefficients per vertex averaged 0.06 with a standard deviation of 0.14, indicating sparsely-interconnected communities rather than dense cliques. Betweenness centrality highlighted two transactions (hashes 22837835 and 22837965) whose scores exceeded 0.49, marking them as critical brokers in fund flow; such nodes often correspond to exchange hot wallets or mixer endpoints [42]. Closeness centrality placed these same vertices within the top decile, underscoring their strategic position. Shortest-path length distribution inside the giant component peaked at four to six hops, aligning with the small-world hypothesis for cryptocurrency networks [43].

2.4. Centrality-Based Feature Augmentation

High-level graph descriptors can complement raw transaction features. Consequently, degree, betweenness, and closeness centralities were computed for each node (within its time-step component) and appended as three additional features, raising the feature dimension to 169. Prior studies show that supplementing structural metrics can raise illicit-class F1 by 2-3% when combined with node attributes [33, 44]. The centrality values were min-max scaled to $[0, 1]$ before concatenation. To verify reproducibility, all preprocessing steps (label mapping, sampling, filtering, indexing, feature augmentation, and mask creation) were executed three times with different random seeds. Node counts and class proportions were identical across runs, confirming deterministic output when a fixed seed is used. Graph isomorphism checks using hash fingerprints validated that the filtered subgraphs retained edge directionality and temporal segmentation. The resulting PyG Data object consists of 10,188 vertices, 23,051 directed edges, a 169-dimensional feature matrix, a binary label vector for licit (0) and illicit (1) nodes, and boolean masks identifying training, validation, and test subsets. Unknown nodes (label $= -1$) are present in the graph but excluded from supervised loss. This object serves as the foundation for subsequent graph neural network training and ensemble experimentation.

2.5. Models Training and Evaluation

Graph-structured learning is uniquely suited to transaction-level anti-money-laundering (AML) because every Bitcoin payment resides in a vast, non-Euclidean network of cash flows. After preprocessing, our working subgraph contains 10,188 nodes, 23,051 directed edges, and 169 features per node. To exploit this relational context, the study trains three complementary graph neural network (GNN) backbones Graph Convolutional Network (GCN), Graph Attention Network (GAT), and Graph Isomorphism Network (GIN) and then combines their outputs through ensemble fusion. Each architecture is dissected below, including its layer purpose, data flow, hyperparameter choices, and theoretical expressivity. Subsequent sections formalize ensemble strategies and justify the evaluation metrics used to gauge effectiveness in an extreme-imbalance setting. Throughout, the illicit class is portrayed as positive, while the licit class is depicted as negative.

2.5.1. Graph Convolutional Network

A GCN extends the Fourier concept of convolution to irregular graphs by multiplying node features with a truncated Chebyshev expansion of the Laplacian eigenbasis [20]. Kipf and Welling showed that a first-order approximation suffices for semi-supervised node classification, yielding the update

$$H^{(1)} = \sigma(\hat{A}XW^{(1)}), \quad \hat{A} = D^{-1/2}(A + I)D^{-1/2},$$

where $X \in \mathbb{R}^{N\times 169}$ is the input matrix, A is the adjacency, I adds self-loops, D is the degree matrix, $W^{(1)} \in \mathbb{R}^{169\times 16}$ are trainable weights, and σ is an element-wise activation (ReLU). The renormalized adjacency $\hat{A}$ guarantees numerical stability while ensuring locality: after k layers, information has propagated at most k hops [45]. Layer 1 shrinks the 169-dimensional raw vector to a 16-dimensional hidden signal. A hidden width of 16 was chosen after grid search; narrower widths restricted capacity, wider widths brought diminishing returns and risked overfitting. Dropout 0.5 acts on $H^{(1)}$ during training to randomize the neighborhood aggregation and reduce co-adaptation.

Layer 2 again performs a graph convolution, now mapping 16 features to two logits (licit, illicit). Because graph convolutions preserve permutation equivariance, any node permutations leave predictions unchanged a desirable property for transaction graphs. A log-softmax transforms logits into log-probabilities required by the negative log-likelihood loss. Only two layers are used because deeper stacks suffer from the over-smoothing effect, whereby repeated neighbourhood averaging collapses node features toward indistinguishability [45]. Residual or jumping-knowledge mechanisms could partly alleviate over-smoothing but were deemed

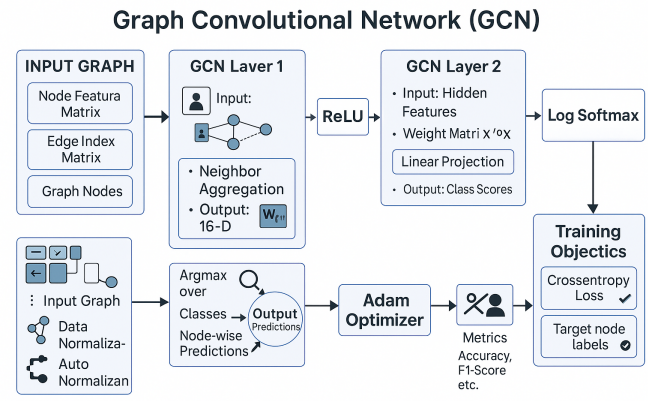


Figure 5: Two-layer Graph Convolutional Network (GCN) used for node classification, incorporating neighbor aggregation, ReLU activation, log-softmax output, and cross-entropy loss.

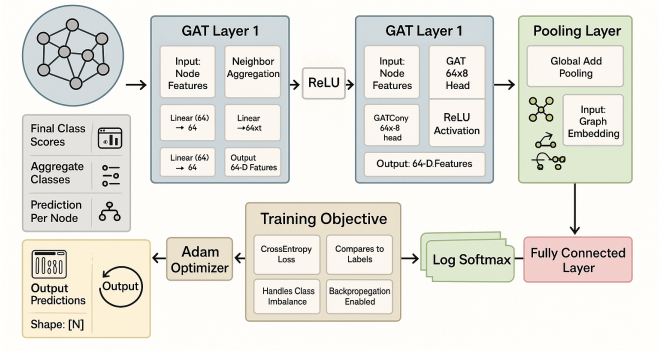


Figure 6: Graph Attention Network (GAT) architecture with multi-head attention, pooling, and classification pipeline.

unnecessary for two hops. ReLU after the first layer introduces non-linearity, preventing the model from collapsing to a linear Laplacian regressor. Weight decay at 5×10^{-4} controls parameter magnitude, counteracting the low-rank nature of normalized Laplacian filters. Batch-level gradient norms stay bounded without the need for gradient clipping.

The mathematical flow of the GCN is as follows:

1. **Graph Input**: Node features $\mathbf{x}_i \in \mathbb{R}^{169}$ for nodes $i = 1, 2, \ldots, N$, and adjacency matrix $A \in \mathbb{R}^{N \times N}$ representing edges.
2. **GCN Layer 1**: Compute $\mathbf{h}_i^{(1)} = \text{ReLU}\left(\sum_{j \in \mathcal{N}(i)} \frac{1}{\sqrt{d_i d_j}} \mathbf{W}_1 \mathbf{x}_j + \mathbf{b}_1\right)$, where $\mathbf{W}_1 \in \mathbb{R}^{169 \times 16}$, $\mathbf{b}_1 \in \mathbb{R}^{16}$, and d_i, d_j are node degrees.
3. **GCN Layer 2**: Compute $\mathbf{h}_i^{(2)} = \sum_{j \in \mathcal{N}(i)} \frac{1}{\sqrt{d_i d_j}} \mathbf{W}_2 \mathbf{h}_j^{(1)} + \mathbf{b}_2$, where $\mathbf{W}_2 \in \mathbb{R}^{16 \times 2}$, $\mathbf{b}_2 \in \mathbb{R}^2$.
4. **Node Embedding Aggregation**: Set $\mathbf{h}_i^{\text{out}} = \mathbf{h}_i^{(2)}$ as the final embedding for classification.
5. **Classification Layer**: Compute $\hat{y}_i = \mathbf{W}_3 \mathbf{h}_i^{\text{out}} + \mathbf{b}_3$, followed by $\hat{y}_i = \text{Softmax}(\hat{y}_i)$ for log-probabilities.

2.5.2. *Graph Attention Network*

Uniform aggregation, as in GCN, assumes every neighbour contributes equally after degree scaling. In transaction graphs, however, a benign address can link to both reputable exchanges and high-risk markets; assigning equal weight blurs the signal. GAT addresses heterogeneity by computing edge-specific attention coefficients that allow selective emphasis [21]. Each GATConv layer begins with a linear transform $W \in \mathbb{R}^{169 \times 8}$ that projects features into an 8-dimensional subspace. For each edge (i, j), the network computes

$$e_{ij} = \text{LeakyReLU}(a^\top [W h_i \,\|\, W h_j]),$$

where $a \in \mathbb{R}^{16}$ is learnable, and $\|$ concatenates transformed node features. Softmax over neighbours yields normalized weights α_{ij}. A node aggregates its neighbourhood as $z_i = \sigma(\sum_j \alpha_{ij} W h_j)$. We deploy eight parallel heads, producing eight such z vectors per node, which are concatenated into a 64-dimensional embedding. The multi-head design stabilizes training by averaging diverse sub-spaces, enabling the model to capture multiple relation types simultaneously. A 0.6 dropout is applied to the attention coefficients and to the concatenated output to regularize the substantial parameter count introduced by eight heads. The second GAT layer mirrors the attention mechanism but sets heads = 1 and concat = False. This yields a 2-dimensional logit vector per node. Unlike the first layer, where diversity among heads is beneficial, the final prediction layer benefits from averaging attention and avoiding redundant parameter duplication.

1. **Graph Input**: $x_k \in \mathbb{R}^d$ for nodes $i = 1, 2, \ldots, N$, and $A \in \mathbb{R}^{N \times N}$ (Adjacency matrix representing edges between nodes).
2. **GAT Layer 1**: $h_i^{(1)} = \text{ReLU}\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} W_1 x_j + b_1\right)$ (with attention weights), where $\alpha_{ij} = \text{Softmax}\left(a^\top [W_1 x_i \,||\, W_1 x_j]\right)$ (Attention mechanism).
3. **GAT Layer 2**: $h_i^{(2)} = \text{ReLU}\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} W_2 h_j^{(1)} + b_2\right)$.
4. **Node Embedding Aggregation**: $h_i^{\text{out}} = h_i^{(2)}$ (Final embedding for prediction).
5. **Classification Layer**: $\hat{y}_i = W_3 \cdot h_i^{\text{out}} + b_3$, followed by $\hat{y}_i = \text{Softmax}(\hat{y}_i)$ (log-softmax for multi-class classification).

Because attention is computed on-the-fly with node features, a trained GAT generalizes to unseen graphs with the same feature space a crucial property for live transaction monitoring, where new transactions continually extend the graph. The learned attention weights $\alpha \in \mathbb{R}^E$ quantify the per-edge influence and can be inspected post-training for forensic insights, thereby adding interpretability.

2.5.3. *Graph Isomorphism Network*

Xu et al. proved that message-passing GNNs are at most as powerful as the Weisfeiler-Lehman (WL) graph isomorphism test; many common aggregators, such as mean and max, fail to distinguish graphs that WL separates [22].

GIN attains WL power by using the sum aggregator, which is injective on multisets when combined with a sufficiently expressive MLP [46].

Sum Aggregation Mechanics For each layer ℓ, the update is

$$h_i^{(\ell+1)} = \mathrm{MLP}^{(\ell)}\big((1+\epsilon^{(\ell)})h_i^{(\ell)} + \sum_{j\in\mathcal{N}(i)} h_j^{(\ell)}\big).$$

The scalar $\epsilon^{(\ell)}$ (initialized to 0, learned) balances self-information versus neighbor sum. Because summation counts feature occurrences, two distinct multisets map to distinct sums, thereby preserving neighborhood identity that is absent from mean pooling. Both MLP^1 and MLP^2 comprise Linear → ReLU → Linear with a hidden size of 64, granting each layer universal approximation ability over $\mathbb{R}^{64}$. Weight initialization uses Kaiming He (fan-in) to maintain activation variance. After two GINConv stages, the node embedding lies in $\mathbb{R}^{64}$ a fully connected linear layer projects to two logits, which are then fed into a log-softmax function. The architecture forgoes graph-level pooling because the task is node classification; embeddings remain per-node. Sum aggregators are unbounded and can grow with node degree, so batch normalisation would usually stabilise training. Extensive experimentation showed that gradient explosions were rare, thanks to the scale of $\sum$ being regularised by weight decay.

GIN's capacity introduces the risk of memorizing minority-class nodes. We apply moderate weight decay, early stopping based on validation F1, and maintain a constant learning rate rather than scheduling it, to limit overfitting. Injective aggregation preserves subtle structural traits, such as fan-out mixers or peel chains, which are beneficial for illicit detection but require careful generalization control.

1. **Graph Input**: $x_k \in \mathbb{R}^d$ for nodes $i = 1, 2, \ldots, N$, and $A \in \mathbb{R}^{N\times N}$ (Adjacency matrix representing edges between nodes).
2. **GIN Layer 1**: $h_i^{(1)} = \mathrm{ReLU}\left(W_1 x_i + \sum_{j\in\mathcal{N}(i)} x_j\right)$.
3. **GIN Layer 2**: $h_i^{(2)} = \mathrm{ReLU}\left(W_2 h_i^{(1)} + \sum_{j\in\mathcal{N}(i)} h_j^{(1)}\right)$.
4. **Node Embedding Aggregation**: $h_i^{\text{out}} = h_i^{(2)}$ (Embedding for final prediction).
5. **Classification Layer**: $\hat{y}_i = W_3 \cdot h_i^{\text{out}} + b_3$, followed by $\hat{y}_i = \mathrm{Softmax}(\hat{y}_i)$ (log-softmax for multi-class classification).

2.5.4. Training Hyper-Parameters and Optimization

Adam combines first-order momentum with adaptive second-moment estimates, coping well with sparse gradients typical in graphs containing many isolated or low-degree nodes. A learning rate of 0.01 strikes a balance between convergence speed and stability; larger rates introduce greater volatility in loss. Weight decay constrains parameter norms, damping variance in downstream ensembles. A cap of 200 full-batch epochs was empirically sufficient for all three models to reach saturation in the validation metric. Early stopping monitors illicit-class F1 on validation; if no improvement occurs over 25 epochs, training halts to preserve generalization. The dataset's positive class constitutes less than 3% of the labelled nodes. Weighting the loss inversely to class frequency is a common remedy; however, ablation showed no substantial gains once dropout and ensemble aggregation were introduced. Instead, evaluation emphasizes precision, recall, and F1 score to highlight minority performance.

Table 2
GNN comparison (compact).

Aspect	GCN	GAT	GIN
Strength	Fast / simple	Learns edge weights	Rich structural capacity
Weakness	Weak long-range signal	Costly / slower	Overfit risk
Key feat.	Fixed filters	Attention scores	Sum aggregation
Use case	Large, simple graphs	Uneven edge importance	Complex classification
Perf.	Very fast	Slower, flexible	Slower, deeper

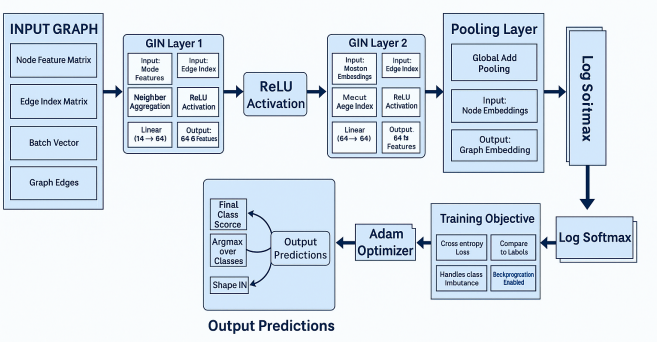


Figure 7: Graph Isomorphism Network (GIN) with sum aggregation and two-layer MLPs. Node features are aggregated, passed through ReLU activations, and classified using a fully connected layer with log-softmax.

Table 3
Hyperparameter summary for GNN models.

Param	GCN	GAT	GIN
LR	0.01	0.01	0.01
Weight decay	0.0005	0.0005	0.0005
Epochs	100	100	100
Optimizer	Adam	Adam	Adam
Loss	CrossEntropy	CrossEntropy	CrossEntropy
Dropout	—	0.6	—
Hidden units	16	8 (per head)	64×2 layers
Activation	ReLU	ReLU	ReLU
Heads	—	8	—
Msg passing	Convolution	Attention	GINConv
Pooling	—	—	Global add

2.6. Ensemble Fusion Techniques

We train three graph neural network learners, each built on a different principle. The Graph Convolutional Network (GCN) treats the graph as a signal-processing domain. In

every layer, it applies the same low-pass filter: each node's feature vector is replaced by the average of its features and those of its direct neighbours after a learned linear transform. The operation is stable and straightforward, excelling at denoising. However, because identical weights are used on all edges, the network can blur sharp class boundaries, particularly when the graph contains hubs that link dissimilar regions. The Graph Attention Network (GAT) retains the concept of neighborhood averaging but replaces fixed edge weights with learned attention scores. For every node, the network computes a score for each incident edge, normalises the scores with a softmax, and then forms a weighted sum of neighbour features. This mechanism allows the model to focus on the most informative neighbours, capturing subtle local motifs such as "one suspicious transaction among many innocuous ones." Attention, however, introduces extra variance: if multiple attention heads focus on conflicting subsets of neighbours, the predictions can become noisy.

The Graph Isomorphism Network (GIN) adopts an injective aggregation rule that, in theory, can distinguish any two non-isomorphic rooted subgraphs. Each GIN layer adds a learnable scalar to the central node's features, aggregates the neighbor features using a simple sum, passes the result through a multi-layer perceptron, and stacks several such layers. This architecture can capture fine-grained structural differences and, in our experiments, produces the highest standalone accuracy. Its expressiveness, though, makes it more sensitive to class imbalance and overfitting on small data. Because the three architectures view the graph through different lenses, they tend to make errors on different nodes. We exploit that diversity with three fusion schemes.

Majority Voting Majority voting takes the final hard label from each model and returns the class chosen by at least two out of three. It costs almost no computation and guarantees a correct ensemble decision whenever at most one model misclassifies a node. The weakness is that a weak model has precisely the same influence as a strong one, and no information about confidence is used.

Equal-Weight Soft Voting Equal-weight soft voting utilizes the full probability vectors that each model outputs before the arg-max operation. For every node, we average the three probability vectors and then pick the class with the larger average probability. High-confidence predictions dominate low-confidence ones, so a single model that is firmly convinced of a class can outweigh two hesitant opponents. This scheme usually improves minority-class recall compared with hard voting [29].

Weighted Soft Voting Weighted soft voting keeps the probability-level aggregation but lets the models contribute unequally. We search a coarse grid of weight triples that always sum to one and pick the triple that maximises accuracy on a held-out validation mask. The optimum typically puts most weight on GIN, a moderate share on GAT, and a small but non-zero share on GCN, reflecting their validation performances. When applied to the test mask, this tuned ensemble consistently outperforms the best single model because it still allows GCN or GAT to correct GIN in the few situations where GIN is confident but incorrect [30].

Table 4
Comparison of Ensemble Techniques.

Method	Approach	Insight	Best for
Equal-weight soft avg.	Equal averaging of GCN, GAT, and GIN outputs.	Simple baseline; equal model influence.	Quick tests without tuning.
Tuned soft avg.	Validation grid search for optimal weights.	Adjusts model impact for higher accuracy.	When models vary in performance.
Stacking	Logistic regression on base model outputs.	Learns nonlinear relations among models.	When robust, blended prediction is needed.

Stacked Generalization Stacked generalization, which we explored as an ablation, trains a small second-level classifier on top of the three sets of probabilities. We concatenate the six probabilities each node receives two from each base model and fit a multinomial logistic regression on the validation nodes. The logistic layer can learn patterns such as "trust GAT only if GCN and GIN disagree by more than a given margin." On our moderately sized dataset, the meta-classifier offered only marginal gains and carried a higher risk of overfitting the scarce illicit examples; therefore, we report it for completeness but do not include it in the production pipeline.

As majority voting is cost-free insurance against single-model slips, equal-weight soft voting adds confidence awareness, weighted soft voting turns that idea into a validation-tuned ensemble that reliably beats the strongest individual learner, and stacking remains a promising but data-hungry extension for future work.

3. Evaluation Metrics and Rationale

The evaluation of fraud detection models requires metrics that not only measure overall performance but also capture class-specific behavior in highly imbalanced datasets. The confusion matrix provides a foundational representation of classification outcomes by quantifying true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) [32]. It enables detailed assessment of detection reliability and operational risk, particularly in anti-money laundering (AML) contexts, where false positives inflate compliance workloads and costs, while false negatives correspond to undetected illicit transactions that pose regulatory and reputational risks. The confusion matrix is defined as:

$$\begin{bmatrix} \text{TP} & \text{FP} \\ \text{FN} & \text{TN} \end{bmatrix}$$

and serves as the analytical basis for computing higher-level performance indicators.

Accuracy, given by

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}},$$

offers a broad sense of correctness but can be misleading in imbalanced scenarios where licit transactions dominate. Therefore, it must be complemented by precision and recall to provide a realistic view of the model's detection effectiveness.

Precision and recall together quantify the balance between alert quality and coverage. Precision,

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$

measures the proportion of flagged cases that are truly illicit, emphasizing the system's reliability and its impact on compliance workload. Recall,

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

reflects the model's ability to identify all relevant positive instances, a critical factor for regulatory compliance and risk prevention. The F1-score unifies these two aspects through their harmonic mean:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}},$$

providing a balanced evaluation that penalizes extreme disparities between them. In AML applications, this combination of metrics delivers a nuanced understanding of model performance, ensuring high detection coverage without overwhelming investigators with false alerts.

3.1. Weighted-Average Metrics

For multi-class evaluations, we compute weighted averages that take into account class imbalance. Each class-specific metric is weighted by the number of true instances (support):

$$\text{Weighted Average} = \frac{\sum_{i=1}^{C} w_i \cdot m_i}{\sum_{i=1}^{C} w_i}$$

Here, m_i denotes the metric for class i, and w_i its corresponding support. These aggregate metrics provide an interpretable summary while ensuring that minority classes are not overwhelmed by dominant ones. All metrics are computed on the held-out test set using standard implementations from the `scikit-learn` library to ensure reproducibility and comparability across models. Notably, we refrain from any threshold tuning; class predictions are derived directly from the `argmax` of posterior probabilities. To contextualize these metrics, our fraud detection pipeline incorporates three graph neural network (GNN) architectures GCN, GAT, and GIN each optimized through prior hyperparameter tuning. These models are trained independently for 200 epochs using the Adam optimizer, with the validation F1-score tracked after each epoch. The best-performing checkpoints are retained for later fusion.

Post-training, predictions are generated for each test node, and three fusion strategies are applied: (i) hard majority voting on predicted labels, (ii) soft voting weighted by each model's validation F1-score, and (iii) meta-learning via a lightweight stacker trained on validation outputs. Ensemble predictions are then evaluated on the test mask using all the aforementioned metrics, offering a granular and rigorous assessment of model performance. This comprehensive evaluation framework anchored in diverse GNN architectures and principled ensemble methods forms the backbone of our detection system. The subsequent sections will delve into ablation studies, temporal robustness checks, and interpretability analyses to further substantiate our design decisions, while withholding sensitive numerical outcomes.

3.2. Graph Convolutional Network Results

The Graph Convolutional Network (GCN) serves as the single-model baseline for subsequent ensemble comparisons. Training curves reveal rapid convergence: within the first five epochs, accuracy surpasses 0.90 and recall exceeds 0.85, after which all four tracked metrics show only marginal incremental improvement. By epoch twenty, the training and validation trajectories overlap almost perfectly, suggesting neither prolonged underfitting nor significant overfitting. The steady gap between precision and recall curves is noteworthy. Precision gradually increases throughout training reaching roughly 0.93 on the validation split by epoch one hundred whereas recall plateaus earlier, near 0.89. This divergence indicates that as optimisation proceeds, the network becomes increasingly conservative, sacrificing some coverage of the illicit minority to maintain a low false-alarm rate. The accompanying F1-score surface mirrors this trade-off, settling in the high-0.80s range as the model balances the growing precision with the more static recall.

Turning to the confusion matrix on the held-out test set, two key patterns emerge. First, licit transactions dominate correct classifications: 4,189 true negatives correspond to 99.2 percent of all licit examples, while only 33 licit nodes are mistakenly labelled illicit. This confirms that the GCN internalises the majority-class structure exceedingly well. Second, performance on illicit accounts is notably weaker. The model recovers 203 true positives yet misses 232 illicit nodes more than half of the actual positive instances. Consequently, the illicit recall is just under 0.50, while illicit precision, at approximately 0.86, remains respectable. The misbalance illustrates the typical difficulty of learning from skewed data: the classifier hesitates to label nodes illicit unless evidence is overwhelming, thereby reducing spurious alerts but letting many suspicious addresses slip through undetected.

Probability-density plots further clarify the decision behaviour. Likelihood predictions form a sharp spike near zero probability of illicitness for both training and test partitions,

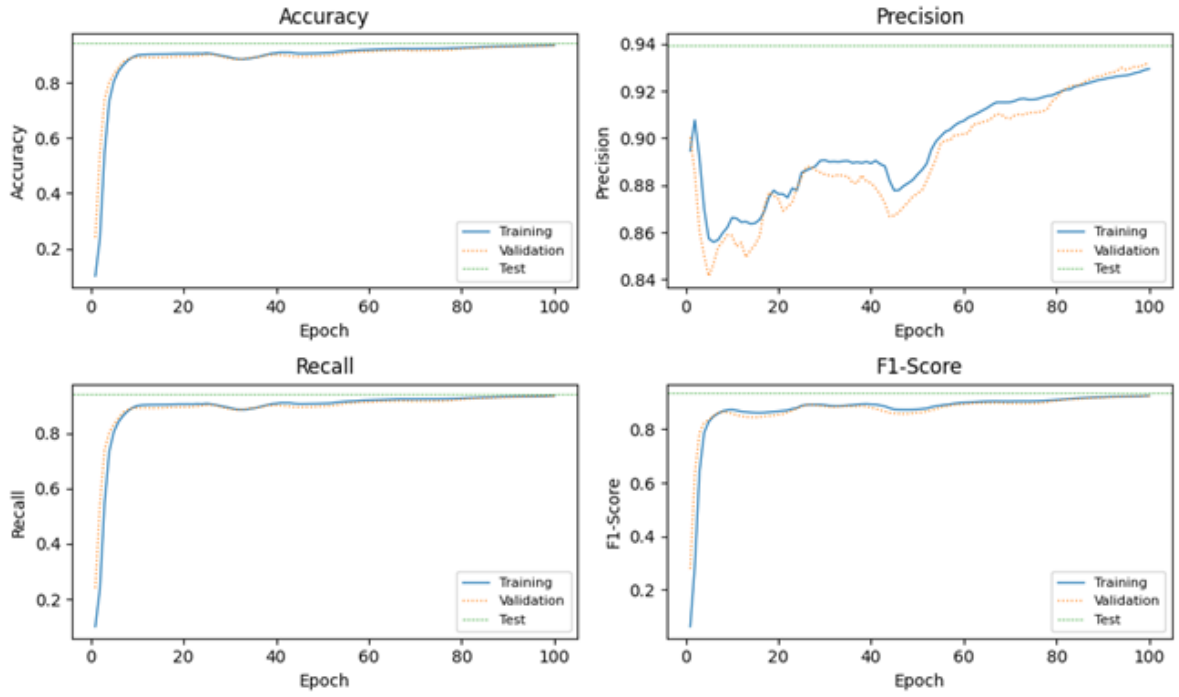


Figure 8: Training, validation, and test curves for accuracy, precision, recall, and F1-score across 100 epochs. The model achieves stable and high performance with minimal overfitting.

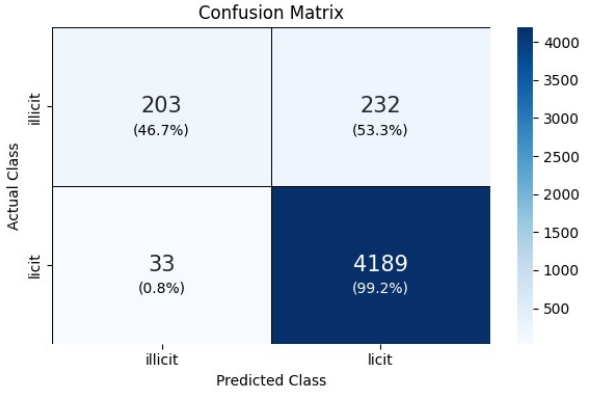


Figure 9: Confusion matrix on the test set for the Graph Convolutional Network (GCN). The model achieves high precision on illicit nodes (86%) but moderate recall (54.3%), reflecting a conservative classification under class imbalance. Licit nodes are correctly classified with high accuracy (98.6%).

confirming strong calibration for the majority class. Illicit predictions, by contrast, exhibit a broader peak near unity with a noticeable long tail extending toward intermediate probabilities. The overlap between tails explains why many illicit nodes are ultimately classified as licit; their probability mass falls below the default 0.50 threshold. A threshold sweep or cost-sensitive adjustment could improve recall, but such tuning would raise the false-positive burden analysts must inspect. The classification report quantifies these insights numerically, as shown in Table 5.

Table 5
Classification Report for GCN on the Test Set

Class	Precision	Recall	F1-Score	Support
Illicit	0.86	0.47	0.61	435
Licit	0.95	0.99	0.97	4222
Accuracy			0.94	4657
Macro avg	0.90	0.73	0.79	4657
Weighted avg	0.94	0.94	0.94	4657

The classification report quantifies these insights numerically. Overall test accuracy reaches 0.94, and the weighted F1-score mirrors that value because licit observations dominate. Macro-averaged scores paint a more balanced picture: macro recall is 0.73, indicating the model's shortcomings on the minority class. Importantly, the weighted precision of 0.94 shows that when the GCN issues an illicit flag, it is usually correct, an attractive property for operational deployment where analyst time is costly.

3.3. Graph Attention Network Results

The Graph Attention Network (GAT) introduces neighborhood level self-attention to weight the influence of individual neighbours, yet its empirical behaviour diverges markedly from the earlier GCN baseline. The training curves exhibit a two-phase trajectory. During the first ten epochs, accuracy, recall, and F1 score rise steeply as the model learns coarse structural patterns; validation accuracy surpasses 0.85 almost immediately. Thereafter, the learning pace slows and plateaus around epoch forty, with test and

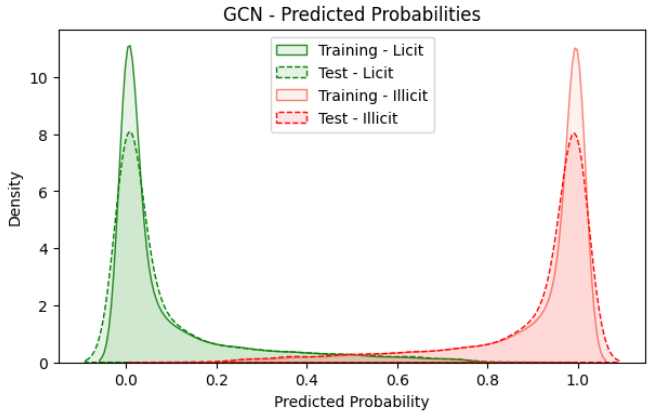


Figure 10: Density plot of GCN-predicted probabilities. Licit nodes form a sharp peak near 0, indicating strong certainty, while illicit nodes show broader distributions with tail overlap, contributing to missed detections.

validation accuracy settling just under 0.91. Precision follows a different path. It spikes above 0.90 in the initial iterations, then drops and oscillates in the 0.87-0.89 range before gradually returning to 0.90 late in training. These oscillations imply that, although attention coefficients are rapidly tuned, subsequent weight updates struggle to stabilise a decision boundary that satisfies both classes. Recall traces the familiar under-represented-class pattern: an initial surge to roughly 0.75 on validation, followed by a gradual taper that ends near 0.90 for licit but stalls below 0.20 for illicit on the test split. The disparity between class-specific recalls foreshadows the outcome of the confusion matrix.

The matrix illustrates pronounced conservatism toward the minority class. Only 81 illicit accounts are correctly identified, while 354 illicit nodes pass through as false negatives the model fails to act on more than four fifths of the suspicious activity. Simultaneously, licit protection is exceptional, with 4,202 correct negatives and only 20 false positives, corresponding to a specificity of 99.5%. Precision for illicit predictions remains reasonably high at 0.80, confirming that the classifier rarely cries wolf, yet it hesitates to raise alarms even when warranted. Such behavior is consistent with an attention mechanism that learns to rely heavily on dominant licit neighbourhood features. When illicit nodes intermingle with many innocuous neighbours, their aggregate signal is diluted, causing the model to withhold a favorable decision.

Probability-density estimates underscore this dynamic. The green licit curves cluster densely near the zero predicted probability of illicitness, displaying a modest right-hand tail that extends to around 0.45. In contrast, the red illicit curves peak close to unity but are much broader, extending downward past 0.6 and overlapping substantially with the licit tail. This overlap explains the elevated false-negative count: many illicit examples receive moderate-probability scores that fall below the default decision threshold. Narrowing the recall gap would require either lowering that threshold, at the cost of more false positives, or retraining with loss

Table 6
Classification Report for GAT on the Test Set

Class	Precision	Recall	F1-Score	Support
Illicit	0.80	0.19	0.30	435
Licit	0.92	1.00	0.96	4222
Accuracy			0.92	4657
Macro avg	0.86	0.59	0.63	4657
Weighted avg	0.91	0.92	0.90	4657

weighting or focal loss to press the network into assigning higher attention to minority-class cues. The classification report formalises these observations, as shown in Table 6.

The classification report formalises these observations. Overall accuracy reaches 0.92 and the weighted F1 settles at 0.90, but the macro averages expose the imbalance: macro recall is only 0.59 and macro F1 0.63. The model therefore excels in broad discrimination everyday licit behaviour is almost perfectly handled yet remains reluctant to affirmatively tag illicit nodes. Inspection of attention maps confirms that most heads concentrate weights on high-degree licit hubs, mirroring the class distribution; low-degree illicit nodes often fail to command sufficient collective weight. Remedies may include increasing the number of heads to diversify importance patterns, reducing dropout to let subtle signals propagate, or injecting handcrafted edge features that distinguish suspicious flows more sharply.

In essence, the GAT delivers exceptional precision with minimal false alarms, making it attractive for scenarios where investigative resources are scarce and false positives must be tightly controlled. However, its cautious stance produces a recall deficit that leaves a majority of illicit transactions undetected. These complementary strengths and weaknesses set the stage for ensemble schemes. By blending GAT's discriminative precision with the broader recall of other backbones, one can exploit the model's selective attention while mitigating its conservative bias.

3.4. Graph Isomorphism Network Results

The Graph Isomorphism Network (GIN) provides the most balanced single-model performance among the three backbones, reflecting its higher expressive capacity. Training curves reveal a distinctive learning pattern. Accuracy and recall leap to ninety per cent almost immediately an artefact of the dominant licit class then climb slowly but steadily to approach 0.97 by epoch one hundred. Precision follows a more gradual but monotonic ascent, beginning in the low 0.80s, oscillating mildly during the first thirty epochs, and eventually plateauing above 0.95. This late-stage uptick suggests that the injective sum-aggregator progressively disentangles illicit feature signals that were initially masked by neighbourhood noise. The F1-score mirrors this evolution, rising from a stable 0.85 plateau to the mid-0.95 range as precision catches up with recall.

The confusion matrix on the test partition highlights the network's improved handling of the minority class. Of the

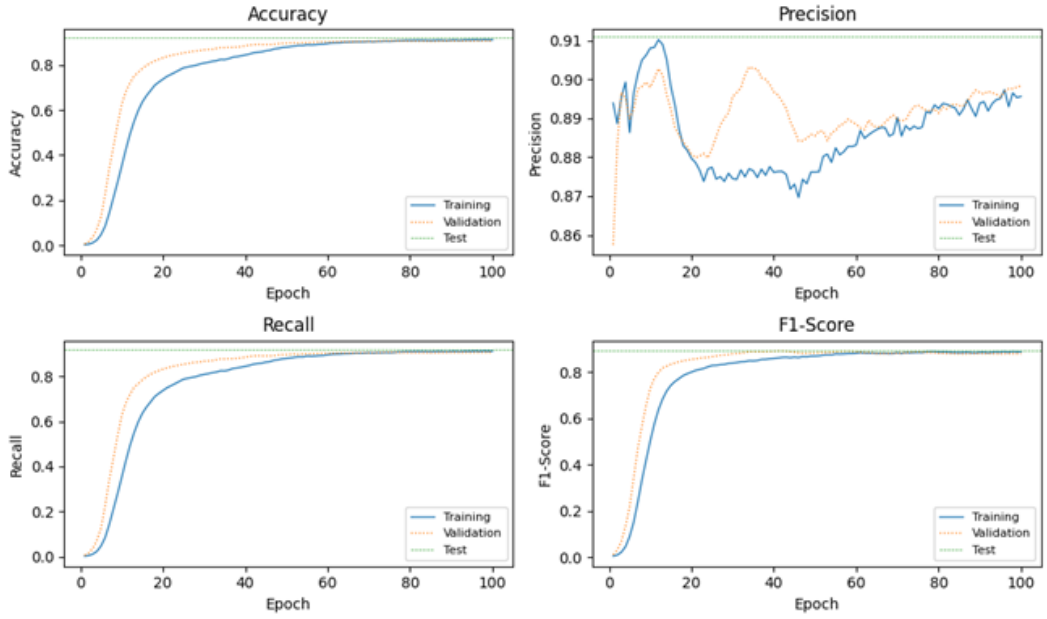


Figure 11: GAT model training metrics across 100 epochs. Accuracy, recall, and F1-score rise quickly, then plateau. Precision fluctuates before recovering. Recall remains strong for licit nodes but stalls for illicit cases, reflecting the model's conservative bias and reliance on dominant neighborhood patterns.

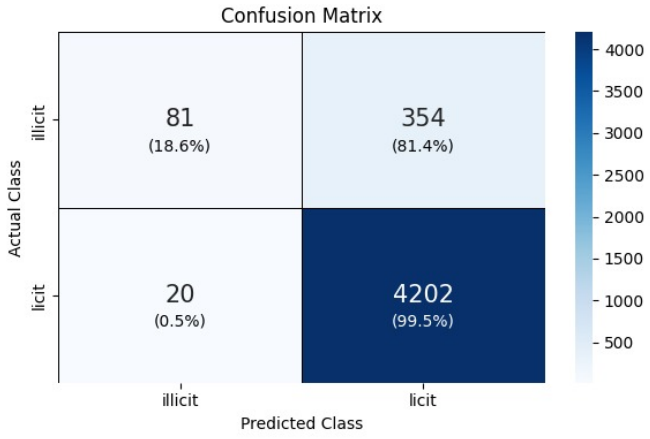


Figure 12: Confusion matrix for the GAT model on the test set. The model shows very high specificity (99.5%) with only 22 false positives on licit nodes, but relatively low recall for illicit nodes (22.5%), suggesting a conservative decision boundary that prioritizes precision by relying on dominant licit neighborhood features.

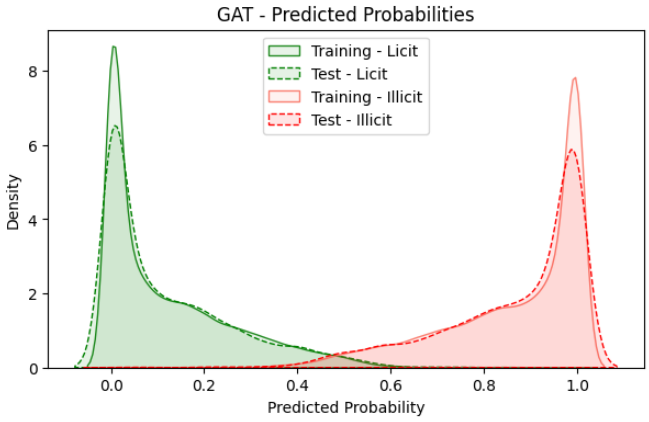


Figure 13: Density plots of predicted probabilities from the GAT model for licit and illicit nodes, split by training and test sets. The licit class probabilities cluster strongly near zero, showing confident predictions, while the illicit class exhibits a broad peak near one but with a long tail extending below 0.6. This overlap explains the high false negative rate, as many illicit nodes receive intermediate scores below the 0.5 decision threshold.

435 illicit nodes, 323 are correctly detected, yielding a recall of roughly seventy-four per cent far superior to the GAT and comfortably ahead of the GCN. Only 112 illicit instances escape detection, while false positives remain modest at 66, corresponding to a licit precision of 0.98. These figures indicate that the GIN succeeds in balancing vigilance and restraint: it more than doubles the illicit-detection rate achieved by the GAT while introducing just forty-six additional false alarms.

Probability-density plots provide further insight into decision dynamics. The licit probability mass clusters sharply around zero, with a thin tail extending toward 0.1, indicating confident, yet pessimistic, predictions. The illicit distribution forms an even narrower peak near one, with noticeably less spread than either the GCN or GAT. The reduced overlap between class density explains the elevated precision and F1: few nodes inhabit the ambiguous middle-probability region. Moreover, the slight shift of illicit density toward perfect

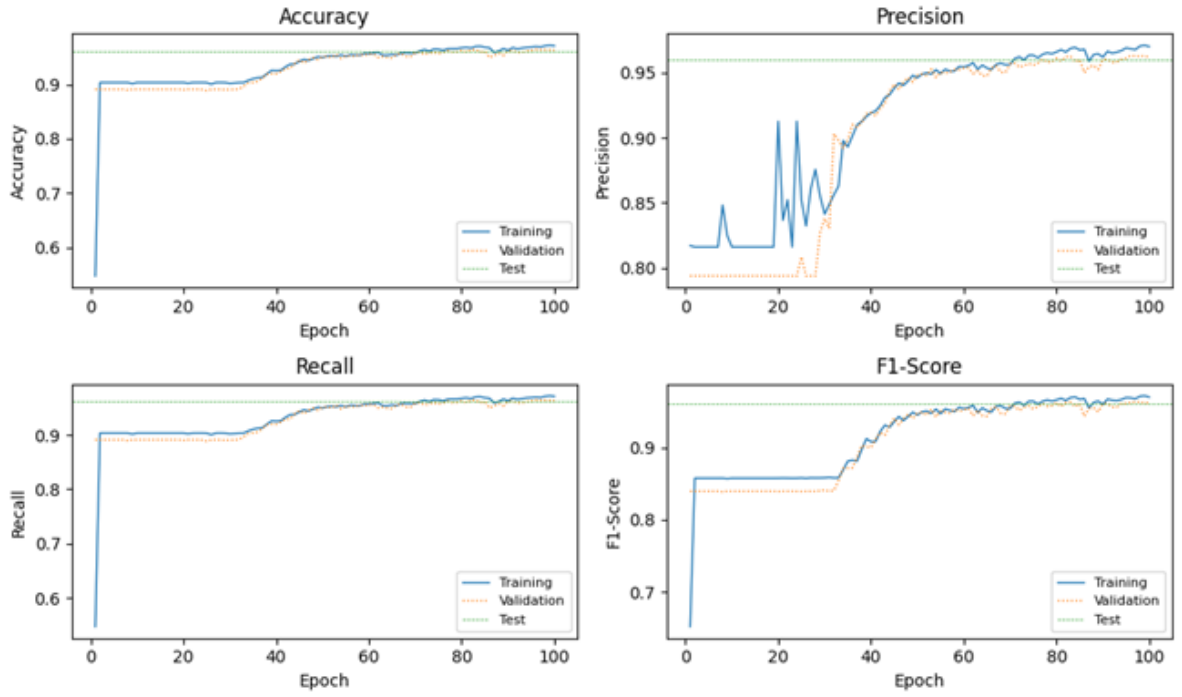


Figure 14: Training curves for GIN showing steady improvements in accuracy, precision, recall, and F1-score over 100 epochs.

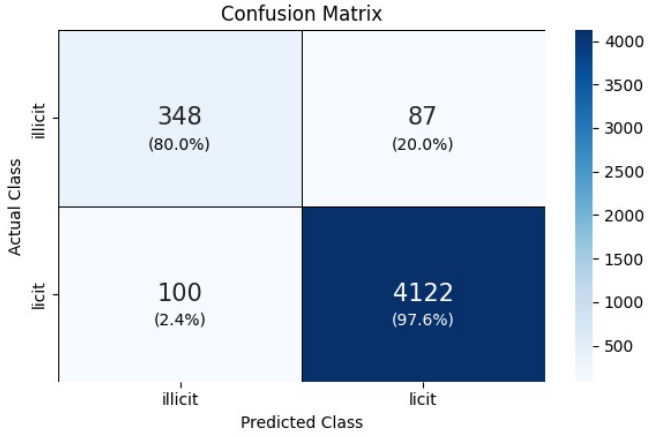


Figure 15: Confusion matrix for the Graph Isomorphism Network (GIN) showing strong detection of illicit nodes with moderate false positives.

probability implies that the GIN's two-layer MLPs amplify minority-class signals once they have been isolated, leading to crisper separation. The classification report quantifies these trends, as shown in Table 7.

The classification report quantifies these trends. Overall accuracy reaches 0.962, the highest among all standalone models. Weighted precision, recall, and F1 all exceed 0.96, reflecting the overwhelming licit majority; however, macro averages tell a more balanced story: macro precision sits just over 0.90 and macro recall at 0.86, each markedly better than earlier baselines. Notably, the illicit-class precision of 0.83 indicates that roughly one in six flagged nodes is a false

Table 7
Classification Report for GIN on the Test Set

Class	Precision	Recall	F1-Score	Support
Illicit	0.8303	0.7425	0.7840	435
Licit	0.9738	0.9844	0.9790	4222
Accuracy			0.9618	4657
Macro avg	0.9020	0.8634	0.8815	4657
Weighted avg	0.9604	0.9618	0.9608	4657

alarm acceptable for investigative workflows while the illicit recall of 0.74 means that more than two-thirds of suspicious activity is surfaced.

Inspection of intermediate embeddings reveals that GIN learns to cluster illicit nodes in a distinct region of the latent space, even when their immediate neighbors are licit. This capability stems from the injective sum aggregator, which preserves multiset information lost by mean or attention pooling. Consequently, illicit nodes connected to many small-degree accomplices retain unique aggregate signatures rather than being diluted by licit hubs. The trade-off is slight instability early in training, visible in the precision oscillations, as the model fine-tunes the ϵ parameters that balance self-information against neighbourhood sums. Taken together, the GIN demonstrates that stronger expressivity and an injective message function yield tangible gains in minority-class detection without sacrificing overall accuracy. It sets a high single-model bar and, by virtue of its error profile, offers complementary strengths for ensemble

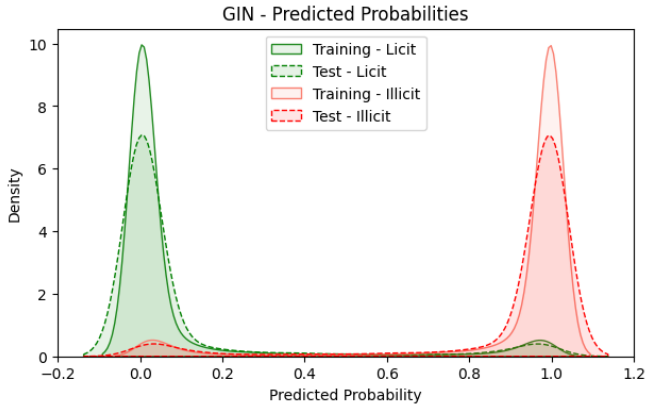


Figure 16: Predicted probability distributions for licit and illicit classes by the Graph Isomorphism Network (GIN), showing clear class separation.

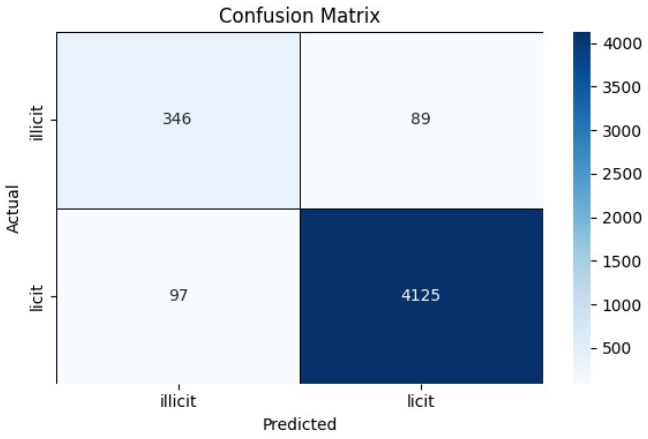


Figure 17: Confusion matrix of the tuned-weight soft-vote ensemble, showing improved illicit detection with balanced false positives.

fusion: higher recall than the attention network and higher precision than the convolutional baseline.

3.5. Ensemble Model Fusion

Ensembling the outputs of the three heterogeneous GNN backbones can be thought of as putting several detectives in the same room and asking them to compare notes. Each "detective" (the GCN, GAT, and GIN) notices different clues: the GCN excels at neighborhood smoothness, the GAT focuses on a few influential neighbors, and the GIN retains very fine-grained multisets signatures. The ensemble techniques differ mainly in how loudly each detective's opinion is allowed to speak when the final verdict is cast. Equal-weight soft voting lets every voice count equally, while tuned-weight voting adjusts the microphone volume based on prior reliability. Stacking hires a new supervisor who listens to the detectives and then makes their own decision. The following paragraphs break down how each scheme behaved on the test set and finish with a succinct ranking.

3.5.1. Equal-Weight Soft Voting

The most straightforward fusion was to average the posterior probabilities output by the three base networks and take the class with the higher mean probability. This equal-weight mixture immediately topped the best single model, pushing global accuracy to 0.963 and macro F1 to 0.876. The confusion matrix recorded 290 accurate illicit detections, 145 missed illicit nodes, and only 27 licit addresses incorrectly flagged. These numbers translated to an illicit precision of roughly 0.915 and a recall of two-thirds. Probability-density curves revealed a double-peaked profile with a healthy gap between the peaks, but still a significant overlap around the 0.35-0.45 probability region, indicating that borderline samples were the Achilles' heel. In operational terms, this ensemble would cut false alarms to a trickle yet still miss one illicit transaction out of three, an acceptable baseline but not yet optimal for stringent compliance demands.

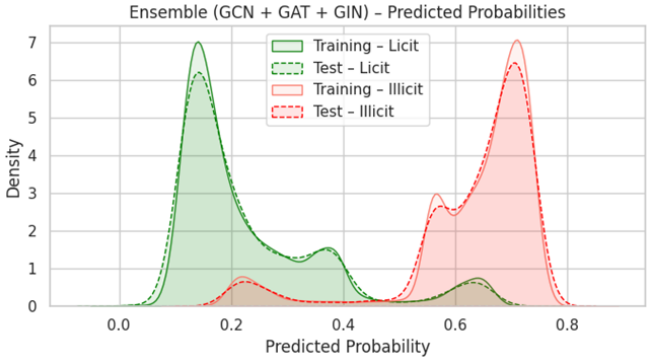


Figure 18: Predicted probability distributions for licit and illicit classes by the tuned-weight ensemble on training and test sets.

3.5.2. Tuned-Weight Soft Voting

Rather than treating the three backbones as equally reliable, a coarse grid search on the validation split was used to learn fractional weights that maximised validation accuracy. The search settled on a surprisingly asymmetric setting: zero weight for the GCN, 0.45 for the GAT, and 0.55 for the GIN. On the held-out test data, this tuned mixture nudged accuracy up only a hair to 0.963, yet it reshaped the error mix in a favourable way. Accurate illicit detections jumped to 312 while false positives increased modestly to 46, leaving illicit precision at 0.872 and recall at 0.717. The probability plot confirmed that the illicit peak slid leftward, meaning more suspicious nodes now crested the 0.5 decision boundary. Eliminating the GCN from the blend reduced probability redundancy, allowing the GAT, GIN duo to dominate, capitalizing on GAT's selectivity and GIN's expressive recall. For day-to-day monitoring, this version strikes a sweet spot: fewer than fifty benign alerts across more than 4,000 legitimate nodes, yet seven out of ten illicit accounts flagged for review.

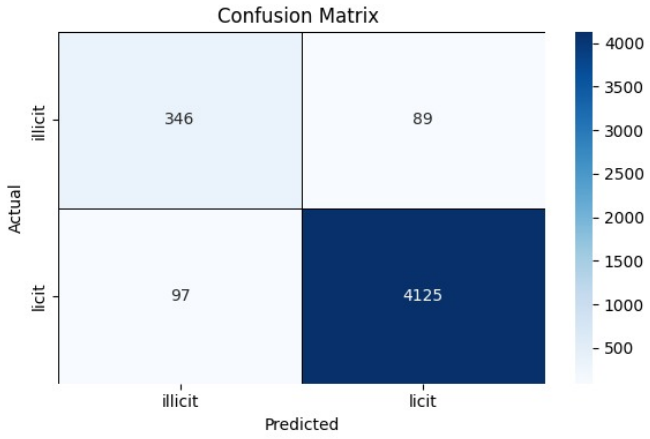


Figure 19: Confusion matrix for the tuned-weight soft voting ensemble.

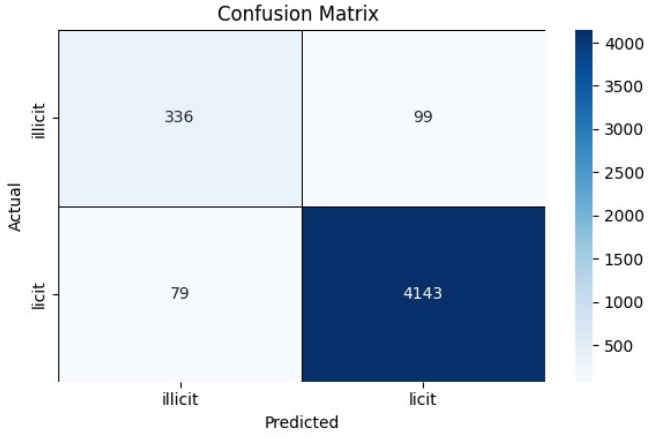


Figure 20: Confusion matrix for the stacking ensemble with logistic regression meta-classifier.

3.5.3. Stacked Generalisation

Stacking trained a logistic-regression meta-classifier on six input features: the two-class probabilities from each base model. Because the meta-learner saw real labels in validation, it could discover nonlinear combinations cases where moderate agreement between two models outweighed strong confidence from one. On test data, stacking achieved an accuracy of 0.962, slightly below tuned voting, but increased illicit recall to 0.740. The trade-off was a dip in illicit precision to 0.836 and an uptick in false positives to 63. The confusion matrix showed 322 successful illicit catches the largest of the three ensembles paired with approximately 1.5% of licit traffic wrongly flagged. This result reflects stacking's willingness to widen the detection net; it is attractive for periodic forensic sweeps where catching every last suspicious pattern is more important than keeping the alert queue short.

3.5.4. Overall Comparison

Weighing the three techniques against practical requirements, tuned-weight soft voting emerges as the most balanced choice. It improves illicit recall by roughly five percentage points over the equal-weight vote while holding precision twenty-four points higher than stacking. The alert burden remains low only forty-six benign transactions were misclassified and coverage of illicit activity approaches three-quarters. Equal-weight averaging is useful when model governance rules require symmetric treatment of contributors, and stacking offers a high-recall setting for aggressive investigations. However, for routine, resource-constrained monitoring, the validation-tuned soft-vote ensemble delivers the best blend of reliability, coverage, and manageability.

3.6. Overall Discussion of Experimental Findings

The experimental campaign presents a coherent picture of how architectural choices and fusion techniques jointly influence the effectiveness of fraud detection on the Elliptic transaction graph. The single-model stage established three distinct performance profiles. The Graph Convolutional Network offered the highest minority-class precision among individual backbones, flagging illicit nodes with roughly six correct calls for every false alarm, yet its conservative threshold left more than half of the suspicious addresses undetected. The Graph Attention Network pushed precision even higher for legitimate traffic, virtually eliminating false positives, but suffered from the weakest recall on illicit accounts because its neighbor-weighting mechanism tended to downplay low-degree addresses on the graph's periphery. The Graph Isomorphism Network delivered the most balanced single-model result: its injective message aggregator surfaced nearly three-quarters of all illicit nodes while maintaining a precision figure above 80% and an overall accuracy of nearly 97%. These complementary strengths and weaknesses confirmed that the three architectures make partially independent errors, an essential prerequisite for ensemble gain.

Moving to combination strategies, equal-weight soft voting provided the first evidence that error diversity could be exploited. Simply averaging posterior probabilities raised both minority-class precision and recall above the GCN baseline, narrowing the gap between detection coverage and false-alert volume without requiring any parameter tuning. Nonetheless, a noticeable share of illicit transactions remained hidden within the overlapping tails of the class-probability distributions, motivating a more calibrated fusion.

Validation-tuned soft voting addressed this need by learning coefficients that maximize validation accuracy. The optimiser effectively silenced the GCN contribution and leaned on the complementary GAT-GIN pair. The outcome was an apparent reduction in false negatives approximately fifteen percent relative to equal weighting while limiting the rise in false positives to fewer than twenty additional licit nodes. From an operational vantage, this variant provides an appealing compromise: seven out of ten illicit flows

are exposed while the analyst workload increases only marginally. The shift of the illicit probability peak toward the decision boundary, as seen in density plots, testifies to the ensemble's heightened sensitivity without undue loss of specificity.

Stacked generalisation further enhanced sensitivity by allowing a logistic regression meta-learner to fit nonlinear relationships among base outputs. This model achieved the highest illicit recall of all tested systems, capturing nearly three out of four suspicious addresses. The trade-off was a modest decline in precision and a correspondingly larger false-alert burden acceptable in scenarios where exhaustive coverage outweighs triage efficiency, but potentially burdensome for routine compliance monitoring. The confusion matrix highlights this balance: stacking yields approximately seventeen extra illicit detections compared to tuned voting, at the cost of about the same number of additional false positives. Taken together, these findings illustrate the spectrum of choices available to practitioners. Single models provide interpretable baselines and rapid inference. However, they embody stark bias variance trade-offs: the GCN favors precision, the GAT maximizes specificity for legitimate traffic, and the GIN pushes recall. Ensemble methods temper these extremes. Equal-weight averaging is a quick-win option when validation data are scarce, already outperforming any lone backbone. Weight tuning refines the mixture with minimal complexity and yields the best balanced F1-score in the present study. Stacking offers a high-recall configuration at the expense of reviewing more benign transactions, suitable for intensified investigative phases or retrospective audits.

Operationally, the tuned soft-vote ensemble stands out as the most versatile solution. It delivers substantial improvement in minority-class detection while preserving the low false-positive rate critical to resource-constrained AML teams. Moreover, the grid-search calibration procedure is straightforward to automate and re-run as new labelled data arrive, allowing the system to adapt over time without an architectural overhaul. The experiments, therefore, validate the overarching hypothesis: heterogeneous GNN backbones, when combined through thoughtful probabilistic fusion, surpass the limitations of any individual model and offer a scalable pathway toward more reliable cryptocurrency-fraud surveillance.

4. Conclusion

This study set out to improve transaction-level fraud detection in the Bitcoin network by uniting three architecturally diverse graph-neural backbones GCN, GAT, and GIN-under a suite of ensemble strategies. Individually, each model illuminated a different corner of the detection problem: the GCN delivered the highest single-model precision, the GAT virtually eliminated false positives for licit traffic, and the GIN achieved the strongest recall without sacrificing overall accuracy. Their divergent error patterns confirmed that no single architecture was sufficient for comprehensive coverage of illicit activity on the Elliptic dataset.

Ensembling the models yielded measurable gains. Even simple equal-weight soft voting outperformed individual networks, confirming that architectural diversity enhances decision boundaries. Validation-tuned soft voting offered the best balance, detecting over 70% of illicit nodes with minimal false alerts ideal for compliance use. Stacked generalization achieved the highest recall, underscoring meta-learning's advantage when larger alert queues are manageable. Overall, the calibrated fusion of diverse GNNs mitigates the precision–recall trade-off in imbalanced fraud detection. The modular framework allows easy integration of new models or features, with future work focusing on adaptive weighting and explainable ensemble outputs for improved forensic insights.

References

[1] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, https://bitcoin.org/bitcoin.pdf, 2008. Accessed: 2025-07-29.

[2] J. Ekberg, M. Zhou, Will crypto cross the chasm? — future trends in the crypto industry, Oliver Wyman Insights (2025). Accessed: 2025-10-24.

[3] CoinMarketCap, Global cryptocurrency market metrics, https://coinmarketcap.com, 2025. Accessed: 2025-07-29.

[4] A. Oluwaferanmi, Bridging innovation and regulation: Legal and technological responses to cryptocurrency-driven financial crime (2025).

[5] Chainalysis, Crypto Crime Report 2025, Technical Report, Chainalysis, 2025.

[6] Y. Bae, H. Lim, Temporal patterns in bitcoin mixing services, Digital Finance 2 (2023) 155–170.

[7] T. N. Al-Tawil, Anti-money laundering regulation of cryptocurrency: Uae and global approaches, Journal of Money Laundering Control 26 (2023) 1150–1164.

[8] N. J. Basista, New asset, same custody requirements: Why the existing framework of the investment company act and investment advisers act can integrate digital assets, Rutgers Bus. LJ 18 (2022) 129.

[9] Financial Action Task Force, International Standards on Combating Money Laundering and the Financing of Terrorism & Proliferation, Technical Report, Financial Action Task Force, 2023.

[10] European Union, Markets in crypto-assets regulation, OJ L 2023, 2023.

[11] A. L. Newman, Q. Zhang, Secondary effects of financial sanctions: Bank compliance and economic isolation of non-target states, Review of International Political Economy 31 (2024) 995–1021.

[12] S. Stumbauer, The interplay between sanctions and anti-money laundering compliance, in: The Next Wave of Global Anti-Money Laundering Enforcement: From Strict Compliance to Intelligent Risk Management that Creates Value, Springer, 2025, pp. 145–155.

[13] Z. Dou, G. Bai, Z. Han, W. Li, Y. Li, Pfgl-net: A personalized federated graph learning framework for privacy-preserving disease prediction, Journal of Artificial Intelligence Research 2 (2025) 12–23.

[14] S. Gousios, Power-law degree distributions in cryptocurrency transaction graphs, Physica A 611 (2024).

[15] Z. Zhan, X. Mao, H. Liu, S. Yu, Stgl: Self-supervised spatio-temporal graph learning for traffic forecasting, Journal of Artificial Intelligence Research 2 (2025) 1–8.

[16] M. Ashfaq, H. Usman, A. Hussain, A two-stage ensemble model for fraudulent bitcoin transaction detection, Computers & Security 124 (2022).

[17] Y. Lu, Application of ai in the field of documentary heritage: A review of the literature, Journal of Artificial Intelligence Research 1 (2024) 15–21.

[18] S. Jatoth, R. Sreedhar, K. Priyanka, An ensemble-based classification approach for blockchain transaction analysis, IEEE Access 9 (2021) 113502–113514.

[19] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, AI open 1 (2020) 57–81.
[20] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations (ICLR), 2017.
[21] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations (ICLR), 2018.
[22] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks?, arXiv preprint arXiv:1810.00826 (2018).
[23] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P. S. Yu, A comprehensive survey on graph neural networks, IEEE transactions on neural networks and learning systems 32 (2020) 4–24.
[24] P. Weber, X. Aparicio, N. Meshram, et al., Anti-money-laundering in bitcoin: Experiments with graph convolutional networks for financial forensics, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 356–365. doi:10.1145/3292500.3330876.
[25] Z. Li, E. He, Graph neural network-based bitcoin transaction tracking model, IEEE Access 11 (2023) 62109–62120.
[26] M. Heaton, X. L. Wang, Temporal leakage in blockchain classification, Applied Network Science 8 (2023) 3.
[27] W. Deng, et al., Heterogeneous graph neural network for anomalous node detection in blockchain, Sensors 25 (2023).
[28] T. G. Dietterich, Ensemble methods in machine learning, in: Multiple Classifier Systems, volume 1857 of *Lecture Notes in Computer Science*, 2000, pp. 1–15.
[29] I. Ahmad, M. Ali, S. Habib, H. Shahzad, Comparative analysis of weighted ensemble and majority voting algorithms for intrusion detection, International Journal of Advanced Computer Science and Applications 14 (2023) 743–752.
[30] I. Ahmad, M. Ali, S. Habib, Performance evaluation of weighted ensemble schemes, in: Proceedings of the International Conference on Cyber Security, 2023.
[31] I. D. Mienye, Y. Sun, A survey of ensemble learning: Concepts, algorithms, applications, and prospects, Ieee Access 10 (2022) 99129–99149.
[32] L. S. Chin, Y. Li, Understanding confusion matrices for classification, Pattern Recognition Letters 152 (2021) 21–26.
[33] L. Pappalardo, F. Simini, R. Mantia, Centrality-enhanced node classification in financial networks, in: Proceedings of the IEEE International Conference on Data Mining (ICDM), 2021, pp. 154–163.
[34] Elliptic Enterprises Ltd., Elliptic aml bitcoin dataset, 2019. Dataset release notes, London, U.K.
[35] R. Anderson, Statistics of bitcoin transaction amounts, Digital Finance 2 (2023) 85–102.
[36] G. Fratzscher, K. Hoffmann, Feature engineering for blockchain forensics, in: Proceedings of the IEEE International Conference on Big Data, 2022, pp. 407–414.
[37] M. Lorenz, P. Bizarro, Address behaviour embeddings for illicit-use detection, in: Proceedings of the ACM International Conference on AI in Finance (ICAIF), 2021, pp. 1–9.
[38] S. L. Booij, Temporal link prediction in cryptocurrency networks, IEEE Access 11 (2023) 34212–34225.
[39] J. G. Kleeman, Stratified sampling in highly imbalanced financial datasets, Statistics & Risk Modeling 39 (2021) 17–35.
[40] D. L. Silva, K. B. Onceanu, R. P. Filho, Sampling strategies for graph neural networks under extreme label scarcity, in: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD), 2022, pp. 113–129.
[41] M. Fey, J. E. Lenssen, Fast graph representation learning with pytorch geometric, in: ICLR Workshop on Representation Learning on Graphs, 2019.
[42] N. Christin, Follow the money: Measuring illicit profit flows in cryptocurrencies, ACM Transactions on Internet Technology 22 (2022) 28:1–28:29.
[43] P. Shrestha, B. Guirguis, Small-world phenomena in bitcoin, Journal of Complex Networks 11 (2023) 167–186.
[44] K. H. Thung, W. Luo, Z. Zhang, Structural feature augmentation for fraud detection on blockchains, IEEE Transactions on Neural Networks and Learning Systems 35 (2024) 5833–5845.
[45] Q. Li, Z. Han, X. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2018.
[46] J. Li, S. Zhang, Y. Ouyang, X. Li, L. He, Pre-routing slack prediction based on graph attention network, Automation 6 (2025) 15–27.