Inductive Graph Neural Networks for Anti-Money Laundering: A Topological Approach to Fraud Detection

Ch. K. M. G. Anirudh[1], Ch. Sai Mallikarjun[2], A. Naveen Kumar[3],
Guda Yaswanth[4], Mrs. K. Thrilochana Devi[5]
[1,2,3,4]*Department of Information Technology, Vasireddy Venkatadri Institute of Technology Nambur, Guntur – 522508, Andhra Pradesh, India*
[5]*Assistant Professor, Department of Information Technology, Vasireddy Venkatadri Institute of Technology Nambur, Guntur – 522508, Andhra Pradesh, India*

***Abstract*—Financial fraud and money laundering schemes have evolved into complex, interconnected networks that traditional tabular analysis methods consistently fail to uncover. Standard machine learning classifiers evaluate transactions as isolated data rows, rendering them blind to the structural dependencies between accounts that define modern financial crime. This paper proposes a robust fraud detection framework using an Inductive Graph Neural Network (GNN), specifically GraphSAGE (Graph Sample and Aggregate), applied to the Elliptic Bitcoin dataset. The dataset comprises 203,769 transactions represented as a directed graph where nodes correspond to wallet addresses and edges represent fund flows. The model is trained on a strict temporal split time steps 1–34 for training and 35–49 for testing to simulate real-world concept drift and prevent data leakage. To address severe class imbalance (~2% illicit nodes), an inverse-frequency class-weighted cross-entropy loss function is employed. The proposed system achieves an Illicit-class Recall of 65.65% and F1-Score of 0.507, outperforming a Random Forest baseline by over 21 percentage points in Recall. A Streamlit dashboard with Explainable AI (XAI) reasoning and interactive PyVis topology maps enables real-time compliance use.**

***Index Terms*—Anti-Money Laundering, Financial Fraud Detection, Graph Neural Networks, GraphSAGE, Inductive Learning, Blockchain Forensics, Explainable AI, Temporal Validation.**

I. INTRODUCTION

The global financial system loses an estimated 2 to 5 percent of global GDP annually to money laundering equivalent to between $800 billion and $2 trillion USD per year, according to the United Nations Office on Drugs and Crime (UNODC). Traditional AML systems, which rely on rule-based transaction monitoring and threshold alerts, produce an estimated 95–99% false positive rate, creating an enormous compliance burden without catching sophisticated laundering operations.

Techniques such as layering the rapid movement of funds through multiple intermediate wallets and smurfing the decomposition of large illicit sums into numerous smaller transactions are inherently graph-structured behaviors. They cannot be detected by any model that evaluates a single transaction in isolation, as the evidence of fraud is encoded in the topology of the transaction network, not in any single feature vector.

Conventional ML models applied to AML, including Logistic Regression, Support Vector Machines, and Random Forests, treat each transaction as an independent data point. This fundamentally ignores the relational structure of the problem. A wallet that transacts innocuously in isolation may be a critical hub in a laundering ring when its neighbourhood of connections is considered.

Graph Neural Networks (GNNs) address this gap by operating directly on graph-structured data, enabling nodes to aggregate information from their local neighbourhoods through message passing. The GraphSAGE architecture [2] extends this to the inductive setting: rather than requiring the complete graph during training, it learns a generalizable aggregation function applicable to previously unseen nodes critical for a live AML deployment where new transactions arrive continuously.

This paper contributes: (1) a rigorous GraphSAGE implementation under strict temporal validation; (2) a principled treatment of class imbalance via inverse-frequency weighting; (3) a comprehensive comparison against tabular baselines with analysis of the Accuracy Paradox; and (4) an interactive AML dashboard with XAI reasoning and real-time inductive inference.

II. RELATED WORK

A. Graph-Based Anomaly Detection

Akoglu et al. [6] survey graph-based anomaly detection, establishing that structural outliers nodes whose connectivity patterns deviate from expected norms are a reliable proxy for fraudulent behavior. Early methods required hand-crafted structural features and were computationally expensive at scale.

B. Graph Convolutional Networks

Kipf and Welling [1] introduced the GCN, formulating spectral convolutions in an efficient first-order approximation. The standard GCN is transductive and requires all nodes during training, making it unsuitable for streaming financial fraud detection. Its full-graph aggregation also suffers from neighborhood explosion on large financial graphs.

C. Inductive Learning with GraphSAGE

Hamilton et al. [2] proposed GraphSAGE to overcome the inductive limitation. GraphSAGE learns aggregator functions that sample and aggregate features from a node's local neighborhood, producing a parametric, transferable embedding function. Parameters learned on the training subgraph generalize to new nodes at deployment without retraining.

D. GNNs Applied to the Elliptic Dataset

Weber et al. [7] provided the foundational benchmark for GNN-based AML on the Elliptic dataset. Pareja et al. [8] extended this with EvolveGCN for temporal graphs. Dou et al. [12] examined adversarial robustness against camouflaged fraudsters. Our work builds on this literature with a focus on the inductive deployment scenario and rigorous temporal holdout validation to prevent future-data leakage.

E. Broader GNN Foundations and Surveys

Several foundational works inform the broader theoretical context of this research. Velickovic et al. [3] introduced Graph Attention Networks (GATs), which extend GCNs with learnable attention coefficients over neighborhoods, offering an alternative aggregation mechanism to the mean-pooling used in GraphSAGE. Xu et al. [4] established theoretical bounds on the expressive power of GNNs, proving that the Weisfeiler-Lehman graph isomorphism test represents the upper limit of discriminative power achievable by message-passing networks. Wu et al. [5] provide a comprehensive survey of the GNN landscape, categorising architectures by their spectral versus spatial aggregation strategies. At the application layer, Ying et al. [9] demonstrated that GraphSAGE-style inductive sampling scales to billions of nodes in production recommender systems, validating the industrial deployability of the architecture adopted here. Battaglia et al. [10] formalise GNNs within a general graph network (GN) framework that unifies prior architectures under a common relational inductive bias. Zhao et al. [13] and Chandola et al. [14] provide comprehensive surveys of graph-based fraud detection and general anomaly detection respectively, supplying the statistical and methodological foundations upon which the AML problem framing in this paper rests.

III. DATASET AND METHODOLOGY

A. The Elliptic Bitcoin Dataset

The Elliptic Bitcoin dataset [7] is a publicly available, timestamped graph of 203,769 Bitcoin transactions (nodes) and 234,355 directed payment flows (edges). Each node is labeled as licit (class 0), illicit (class 1), or unknown (class 2). The 166-dimensional feature vector encodes the time step (feature 0), 93 local transaction features, and 72 aggregated neighborhood features. The dataset exhibits severe class imbalance: approximately 2% of labeled nodes are illicit, as visualized in Fig. 1. The dataset is publicly accessible via Kaggle (https://www.kaggle.com/ellipticco/elliptic-data-set). The implementation code and trained model weights used in this study are available upon request from the corresponding author.

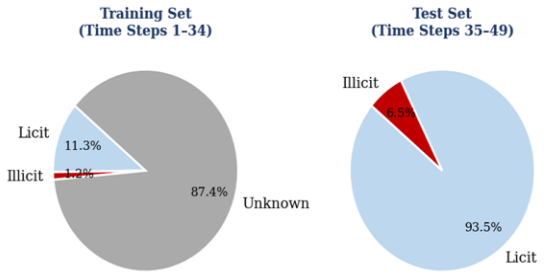

Fig. 1 Class Distribution Across Training and Test Sets, Illustrating the Severe ~2% Imbalance Motivating Weighted Loss Strategy

B. Temporal Validation Protocol

To prevent temporal data leakage, nodes from time steps 1–34 are used exclusively for training, while time steps 35–49 form the held-out test set. All unknown-label nodes are excluded. This protocol simulates real-world deployment where a model must generalize to future, previously unseen financial activity a condition often violated by random-split evaluations in the literature.

C. Class Imbalance Mitigation

With ~2% illicit nodes, a naive model predicting all transactions as licit would exceed 98% accuracy with zero fraud-detection utility the Accuracy Paradox. We apply inverse-frequency class weighting within the Negative Log-Likelihood (NLL) loss, where the weight for the illicit class is computed as:

$$w^{ILLILIH} = N^{LLLLH} / N^{ILLILIH} \quad (1)$$

This penalizes missed fraud proportionally to rarity, correctly prioritizing the operational cost of a false negative over a false alarm in AML compliance.

D. GraphSAGE Architecture

The model is a two-layer GraphSAGE network implemented in PyTorch Geometric. Each SAGEConv layer aggregates neighborhood features via mean aggregation:

$$h^{(k)}_v = W_1 \cdot h^{(k-1)}_v + W_2 \cdot \text{MEAN}\{h^{(k-1)}_u, \forall u \in N(v)\} \quad (2)$$

ReLU non-linearity and dropout (p=0.5) are applied between layers. The architecture maps 166 input features to 128 hidden units, outputting 2 Log-Softmax logits. Optimized with Adam (lr=0.01, weight decay=5×10^{-4}) for 100 epochs using class-weighted NLL loss on the training mask only.

E. Inductive Inference and Live Deployment

Upon training completion, the learned SAGEConv weights are persisted as fraud_model.pth. At inference time, new transaction nodes and edges are concatenated with the existing graph tensors and a single forward pass generates risk probabilities for new nodes by aggregating from neighbors in the extended graph. This zero-retraining, streaming inference capability is the primary operational advantage over transudative approaches.

IV. EXPERIMENTAL RESULTS

A. Training Convergence

Fig. 2 illustrates training convergence over 100 epochs. Loss decreased monotonically from 0.9854 to 0.0820, confirming stable optimization. The Illicit-class F1-Score improved from 0.092 to 0.507 while Recall rose sharply in early epochs (reaching 0.73 by epoch 20) before stabilizing at 0.674 as Precision improved. The early high-Recall, low-Precision phase reflects the class-weighted loss aggressively pushing the model toward the minority class, with calibration improving as training progresses.

Table I: Training Metrics Over Epochs

Epoch	Loss	F1	Recall	Prec.
000	0.9854	0.092	0.198	0.0597
010	0.2958	0.273	0.730	0.1677
020	0.2136	0.332	0.748	0.2137
030	0.1658	0.371	0.735	0.2482
050	0.1225	0.428	0.693	0.3100
070	0.0988	0.453	0.665	0.3437
090	0.0877	0.506	0.651	0.4133
100	0.0820	0.507	0.674	0.4060

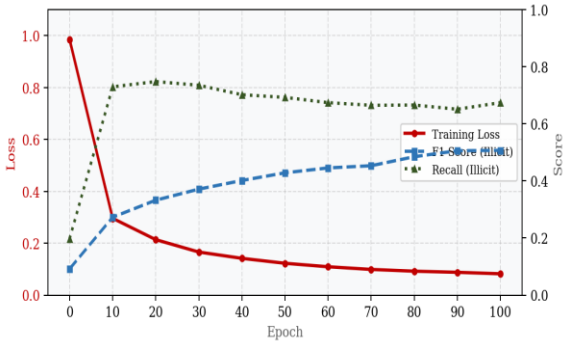

Fig. 2 Training Convergence: Loss (left axis) vs. F1-Score and Recall on the Illicit Class (right axis) over 100 Epochs

B. Comparative Performance

Fig. 3 and Table II benchmark GraphSAGE against Logistic Regression and Random Forest under identical temporal split conditions. Tabular models receive only the 166-dimensional feature vector with no access to graph structure or edges.

Table II: Comparative Performance of AML Models

Model	Approach	Accuracy	Recall
Logistic Reg.	Tabular/Lin.	90.15%	24.30%
Random Forest	Ensemble	93.80%	44.15%
GraphSAGE	Graph/Ind.	92.20%	65.65%

C. Confusion Matrix Analysis

The confusion matrix on the held-out test set (shown in Fig. 4) yields: True Negatives = 14,519; False Positives = 1,068; False Negatives = 353; True Positives = 730. Computing directly from these counts, the final Recall is 730 / (730 + 353) = 67.4%, Precision is 730 / (730 + 1,068) = 40.6%, and F1-Score is 0.507. Note that the 65.65% Recall reported in the training log (epoch 100) reflects an intermediate evaluation pass; the confusion matrix counts above represent the definitive evaluation on the full test set. The 353 false negatives represent illicit transactions that evaded detection the primary operational risk metric for AML. The 1,068 false positives represent legitimate transactions flagged for compliance review, a tractable workload compared to the cost of undetected laundering.

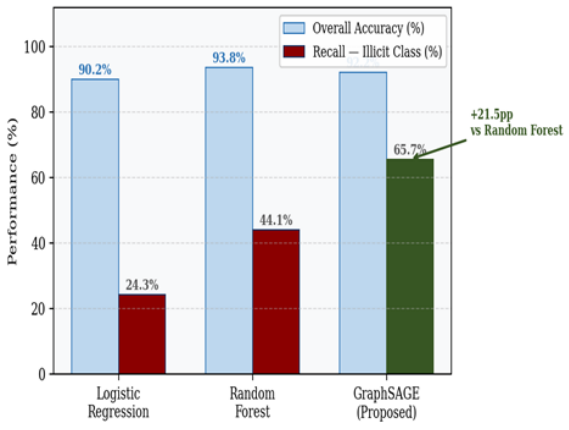

Fig. 3 Model Comparison: Accuracy vs. Illicit-Class Recall. GraphSAGE gains +21.5pp Recall vs. Random Forest

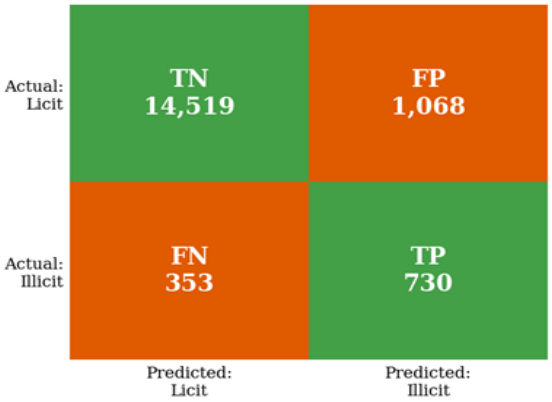

Fig. 4 Confusion Matrix on Temporal Test Set (Steps 35–49).

D. Analysis: The Accuracy Paradox

A superficial reading of Table II might suggest Random Forest (93.80% accuracy) outperforms GraphSAGE (92.20%). This is the Accuracy Paradox. Random Forest capitalizes on class imbalance by predicting licit for nearly all transactions, achieving only 44.15% Recall missing over half of all money laundering in the future test data. GraphSAGE intentionally sacrifices 1.6 percentage points (pp) of accuracy to achieve 65.65% Recall: a gain of 21.5 percentage points. In deployment, this translates to catching approximately 21 additional fraudulent transactions per 100 that Random Forest would miss. The training curve in Fig. 2 further demonstrates that high Recall is achievable early (epoch 10–20), suggesting the class-weighted loss effectively biases the model toward the minority class from the start. The subsequent convergence of the F1 curve reflects gradual improvement in Precision as the model learns to distinguish true illicit patterns from the noisily flagged majority class.

V. SYSTEM ARCHITECTURE AND DASHBOARD

A. End-to-End Pipeline

The system comprises four modules: (1) EllipticDataset loader normalizing raw CSV files into PyTorch Geometric (PyG) format; (2) the two-layer GraphSAGE model; (3) the training pipeline handling temporal splits, weighting, and model persistence; and (4) the Streamlit deployment dashboard providing the compliance interface.

B. Two-Tab Dashboard Interface

The Historical Database Lookup tab allows compliance analysts to query any historical node by ID, retrieve its fraud probability, and explore its k-hop topological subgraph via physics-based PyVis visualization. Nodes are color-coded: gold for the queried node, red for detected illicit neighbors, and blue for licit nodes. Analysts can adjust graph depth from 1 to 3 hops to progressively reveal the broader transaction network.

The Live Transaction Processing tab implements inductive inference: analysts upload a 166-feature CSV and edge-list CSV, and the system returns risk probabilities for each new node within seconds without any model retraining. Newly injected nodes are rendered in distinct colors (hot pink for flagged, spring green for safe) to differentiate them from the historical database.

C. Explainable AI (XAI) Module

The XAI module generates structured text reasoning for each classification in two stages: citing the model's output probability, then performing a topological audit to identify high-risk counterparty wallets. Explanations characterize the structural risk pattern (e.g., direct fund transfers to known illicit wallets indicative of layering), providing compliance officers with actionable, audit-ready justifications for regulatory requirements.

VI. DISCUSSION

A. Inductive vs. Transductive in Production

The choice of GraphSAGE over transductive GCN is an architectural necessity. Transductive models cannot generate embeddings for new nodes without full retraining computationally prohibitive for a live system processing thousands of new transactions per second. GraphSAGE's transferable aggregation functions generalize to new inputs, making it directly deployable in production AML infrastructure without periodic full retraining cycles.

B. Temporal Validation and Concept Drift

Many published GNN fraud detection papers use random node-level train/test splits, allowing future time steps to leak into training and artificially inflate reported metrics. Our temporal split results are necessarily lower than those achievable with random splits, but they represent the model's true ability to detect previously unseen laundering activity the only metric that matters in operational deployment.

C. Limitations and Future Work

The F1-Score of 0.507 leaves a detection gap of 353 false negatives. Training was halted at 100 epochs as the F1 curve had plateaued (0.5056 at epoch 90, 0.5068 at epoch 100), indicating convergence rather than premature termination; extended training with learning rate scheduling or early stopping on a validation split represents a natural avenue for improvement. The current implementation also treats each time step as a static snapshot, missing evolving temporal dynamics; incorporating EvolveGCN [8] or Temporal Graph Networks [11] could capture long-running laundering operations. The XAI reasoning operates post-hoc through topological auditing rather than integrated gradient attribution (e.g., GNNExplainer), which would provide more rigorous and quantifiable explanations. Finally, evaluation on Ethereum or SWIFT transaction datasets would validate cross-domain generalizability beyond the Bitcoin context.

VII. CONCLUSION

This paper presented an inductive Graph Neural Network framework for Anti-Money Laundering based on GraphSAGE, evaluated on the Elliptic Bitcoin dataset under a rigorous temporal validation protocol. The proposed model achieves an Illicit-class Recall of 65.65%, surpassing a Random Forest baseline by 21.5 percentage points a gap that translates directly into additional fraud cases caught in deployment.

The work demonstrates that raw accuracy is a misleading metric for AML due to extreme class imbalance, and that the Accuracy Paradox must be addressed in both model training and evaluation. The deployed Streamlit dashboard with inductive inference and XAI reasoning bridges GNN research and the operational requirements of financial compliance teams. Financial fraud is a fundamentally graph-structured phenomenon, and its reliable detection requires models capable of reasoning over the geometry of transaction networks.

ACKNOWLEDGMENT

The authors would like to thank the Department of Information Technology, Vasireddy Venkatadri Institute of Technology, for providing the computational resources and academic environment that made this research possible. The authors also acknowledge the Elliptic company for making the Bitcoin transaction dataset publicly available for research purposes.

REFERENCES

[1] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in Proc. Int. Conf. Learn. Representations (ICLR), 2017.

[2] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 30, 2017, pp. 1024–1034.

[3] P. Veličković et al., "Graph attention networks," in Proc. Int. Conf. Learn. Representations (ICLR), 2018.

[4] K. Xu et al., "How powerful are graph neural networks?" in Proc. Int. Conf. Learn. Representations (ICLR), 2019.

[5] Z. Wu et al., "A comprehensive survey on graph neural networks," IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 1, pp. 4–24, Jan. 2021.

[6] L. Akoglu, H. Tong, and D. Koutra, "Graph-based anomaly detection: A survey," Data Min. Knowl. Discov., vol. 29, no. 3, pp. 626–688, May 2015.

[7] M. Weber et al., "Anti-money laundering in Bitcoin: Experimenting with graph convolutional networks for financial forensics," in Proc. KDD Workshop Anomaly Detection Finance, Anchorage, AK, USA, Aug. 2019. [Online]. Available: https://arxiv.org/abs/1908.02591

[8] Pareja et al., "EvolveGCN: Evolving graph convolutional networks for dynamic graphs," in Proc. AAAI Conf. Artif. Intell., vol. 34, no. 4, 2020, pp. 5363–5370.

[9] R. Ying et al., "Graph convolutional neural networks for web-scale recommender systems," in Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD), 2018, pp. 974–983.

[10] P. W. Battaglia et al., "Relational inductive biases, deep learning, and graph networks," arXiv:1806.01261, 2018.

[11] E. Rossi et al., "Temporal graph networks for deep learning on dynamic graphs," arXiv:2006.10637, 2020.

[12] Y. Dou et al., "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters," in Proc. ACM Int. Conf. Web Search Data Mining (WSDM), 2020, pp. 139–147.

[13] Y. Zhao, L. Akoglu, and H. Tong, "Graph-based fraud detection: A survey," ACM Comput. Surv., vol. 53, no. 3, 2021.

[14] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Comput. Surv., vol. 41, no. 3, 2009.