

Graph Neural Network-Based Financial Fraud Detection and Cybersecurity Risk Analysis in Banking Networks

Md. Sumsuzzaman ^{1*}, Md Abdullah Enayet ², Sana Ullah ³, Masrufa Tasnim ⁴, Mst Anupa Nasrin Khan ⁵, Tanaya Jakir ⁶, Shakila Sultana ⁷ and Arat Hossain ⁸

¹ Master's of Business Administration, International American University, USA.

² Master of Science in Cybersecurity, Washington University of Science and Technology, USA.

³ MS in Management – Business Analytics, St. Francis College, Brooklyn, New York

⁴ Master of Business Administration (MBA) – Human Resource Management, University of Dhaka, Bangladesh.

⁵ Master of Science in Zoology (Entomology Focus), The University of Rajshahi, Bangladesh.

⁶ PhD in Information Technology, University of the Cumberland, USA.

⁷ Master's in Accounting (M.A.), National University, Bangladesh.

⁸ PhD in Information Technology
University of the Cumberland, USA.

Abstract

Existing modern banking networks are vulnerable to two classes of threat; typically scoped separately: (a) financial fraud and money laundering (the transaction layer), and (b) cyber intrusions on the network-traffic layer. The two threats are relational, yet most detectors flatten this structure to per-record tabular features or use node-centric GNNs that neglect the wealth of attributes carried by transactions and flows. We propose a unified edge-centric GNN framework which models both fraud detection and intrusion detection as the same fundamental problem: binary edge classification on a directed multigraph. Trained on two strongly dissimilar graphs: an account-to-account money-laundering graph with 515,088 account nodes and 5,078,345 transaction edges (real data), and a NetFlow-based intrusion (derived from NF-UNSW-NB15-v2). A single backbone is trained to be able to learn on both graphs by fusing edge features directly into neighborhood aggregation. We compare the proposed model to strong tabular baselines (logistic regression, random forest, XGBoost and a multilayer perceptron) as well as structure-only GNNs (plain GCN and GraphSAGE) across three random seeds. Edge-feature fusion is decisive: for the money-laundering graph it raises area under the precision-recall curve (AUPRC) from approximately 0.004 for structure-only GNNs to 0.166, and on the intrusion graph the proposed model achieves both best AUPRC (0.985) and highest precision (0.958) of any method evaluated. We additionally provide graph-native explanations through Integrated Gradients and edge-mask attribution along with a node level risk-scoring layer to transform edge probabilities to interpretable Low, Medium, and High-risk tiers. Results are reported transparently, including settings in which even extremely well-tuned tabular models remain competitive on this extreme-imbalance money laundering task, and we discuss the practical and regulatory value of a single relational, interpretable model over both banking threat surfaces.

Keywords: graph neural networks; financial fraud detection; anti-money laundering; network intrusion detection; explainable artificial intelligence; banking cybersecurity; class imbalance.

1. Introduction

Banks run two coats of tightly coupled but separately defended armor. The financial surface, where money moves between accounts, and where fraud, money laundering and account takeover create a direct loss from theft of cash or goods. The second is the network perimeter, where hosts and services are in communication for exchanging traffic and where intrusions, scans and denial-of-service attacks pose threats to availability and confidentiality. These surfaces are monitored and protected by

separate teams with different tools, even though each compromise that begins at the network layer almost always will impact the financial layer, and a laundering ring often orchestrates actions only recognized when transactions are viewed as a connected whole rather than as individual rows.

The two problems share a common property. Both are naturally graphs. In the financial surface, accounts are points, and every transaction is a directed edge bearing attributes like amount, currency and payment channel. The network surface consists of nodes (hosts or services) and directed edges that represent each traffic flow, capable of carrying dozens of NetFlow statistics. In both cases the edge is suspicious: a laundering transfer, or a harmful flow. This observation motivates a unified approach to both tasks, i.e. the binary classification of edges on a directed multigraph. This formulation is well-suited for graph neural networks since they learn representations respecting connectivity [1, 2, 3, 5]. The literature has, however, two gaps that keep on being present. Xiwang: The problem arises from twofold: first, since many GNN detectors are node-centric, edge attributes such as weights and types only serve as second-class information which discards precisely the signal that draws the line between a normal transfer and a laundering transfer, or between benign flow and attack flow. Second, financial-fraud GNNs [9, 10, 11, 12] and intrusion-detection GNNs [6, 7] have emerged as two independent literatures so that no single model is yet able to transfer across both banking threat surfaces. This paper addresses both gaps.

The contributions of this work are three-fold. (1) We restate the detection of financial fraud and cyber intrusion in banking as a single, directed multigraph edge classification task and approach it with one edge-centric graph neural network backbone that fuses edge features into message passing. (2) Under a memory-efficient neighbor-sampling training suitable for a single consumer GPU, we evaluate the framework on two structurally diverse real graphs: an account-transaction money-laundering graph and a NetFlow intrusion graph. (3) We measure the benefit of edge-feature fusion through a structural-precision ablation, an asymmetrical tabular baseline comparison, imbalance-aware metrics, and three seeds. Graph-Native Explainability: (4) enables integration of graph-native explainability through Integrated Gradients and edge-mask attribution, alongside a node-level risk-scoring layer that yields interpretable risk tiers for analysts. (5) We report results faithfully, even where tabular models are still even competitive so that the strengths and limits of the approach are unmistakably identifiable.

2. Related Work

2.1 Graph neural networks

Graph convolutional networks introduced an efficient layer-wise propagation rule for semi-supervised node classification [1]. GraphSAGE extended this idea to an inductive setting, where sampling and aggregating from a node neighborhood is crucial for large growing graphs [2]. Graph attention networks accommodate weighting neighbors adaptively [3], while the graph isomorphism network described the representational limits of message passing [4]. These directions are summarized in a broad survey [5]. Banking graphs are very big and change over time, so models based on the inductive, neighbor-sampling paradigm of GraphSAGE can be used as our backbone.

2.2 Graph learning for financial fraud and money laundering

Prior approaches, focusing on anti-money laundering for cryptocurrency ledgers, already found relational features increase performance of forensic classification [11, 12]. Later GNN detectors have explored two obstacles that pervade financial fraud: the concealment of a large cohort of opportunistic models, solved by camouflage-resistant neighbor selection [9] and high class imbalance, dealt with label-balanced sampling and neighbor choice [10]. Naive GNN aggregation under manipulation [19] has also been directly analyzed. When it is not possible to access logs of real transactions, synthetic transaction simulators such as PaySim provide labeled data [18]. These are node-centric, single-domain methods, while our work is edge-centric and evaluated across both the financial and network surface.

2.3 Graph learning for network intrusion detection

To use edge features and topology jointly for the purpose of intrusion detection, E-GraphSAGE modified GraphSAGE for flow edges and achieved strong performance on flow-based benchmarks [6]. The NF-UNSW-NB15-v2 dataset applied to here falls in a family of convertible NetFlow datasets created with the aim of interoperability between network-security data across environments [7], and is based on the well-known UNSW-NB15 testbed [8]. While our backbone for the intrusion is E-GraphSAGE-style, it differs only in that this isn't even about the backbone alone; rather, it's stating that one such backbone covers money laundering and intrusion all in a single banking-oriented container.

2.4 Explainability and risk scoring

An analyst must look at the reason that an alert fired for a detector to be actionable on a regulated bank. Integrated Gradients explains a prediction by summing gradients through paths from input features [17] while GNNExplainer identifies key subgraph and features in the prediction of a graph [16]. We employ Integrated Gradients for edge-feature attribution, and adopt an edge-mask procedure inspired by GNNExplainer for local subgraph explanations: calibrated edge probabilities are pooled into node-level risk bands.

3. Datasets and Problem Formulation

3.1 Unified edge-classification formulation

We model a banking surface as a directed multigraph $G = (V, E)$ where V is the set of nodes that represent entities (an account, or a host/service) and E is the set of nodes which represent interactions (a transaction or network flow). Each edge is labelled with a feature vector and a binary value (1 for illicit lazy transfer or attack flow, 0 otherwise). The goal here is therefore to learn a function that given the pair, allows us to predict this label for unseen edges. The use of this single definition encompasses both banking surfaces and is what covers both a model as well as training procedure, enabling it all to be shared.

Formally, given a graph G with node set V and edge set E , the goal is to learn a parameterized function that assigns each edge an illicit probability:

$$\hat{y}_e = f_\theta(x_e, G) = P(y_e = 1 \mid x_e, G), \quad e \in E \quad (1)$$

3.2 Money-laundering transaction graph

The financial embedding employs a small transaction split generated by an AMLSim-style simulator (which we refer to as HI-Small) based on real-world data in account-to-account money-laundering situation. There are two components here: An accounts are nodes and a transaction is directed edge. The edge features consist of the amounts paid and received, payment currency and receiving currency, payment type (SWIFT or SEP), and a time delta indicating the number of seconds since last transaction with source account. The graph has 515088 account nodes and 5078345 transaction edges. Thus, this is one extreme imbalance setting with 0.10 percent of the edges as labelled launderings. Since transactions are timestamped, a chronological sorting of edges and partitioning the entire graph into training-validation-test sets at fractions 70, 15, and 15 prevents information from filtering forward.

3.3 NetFlow intrusion graph

We utilize the NF-UNSW-NB15-v2, where it is a standardized version of NetFlow dataset with 43 flow features which contains approximately 2.39 million flows (3.78 per cent are attacks). The flow is represented by a directed edge and has the 43 NetFlow statistics as edge features. Limitation on node identity. The copy of NF-UNSW-NB15-v2 that we obtained was stripped of the host-identifying columns (source and destination IPv4 addresses), so a literal host-to-host topology could not be reconstructed. Thus we define node identity to be a transparent proxy: an (port bucket, protocol) pair. This is why well-known destination ports below 1024 retain their precise value, as they collate service traffic, while higher ports simply zero into fixed-width buckets so that ephemeral client ports are pooled by broad role rather than treated uniquely. The resulting learnable scale-free client-role to service graph is 744 pseudo-hosts. We note here specifically that flows from different physical

hosts using the same port and protocol collapse onto a single node, while a single host using many ports spreads across multiple nodes. Hence the validation of our results on the intrusion dataset shows that your shared edge-classification architecture generalizes to different graph types, not literal correctness of IP-level attribution. We perform a stratified random split on the label, since this dataset has no reliable per-flow timestamp.

Table 1. Summary of the two banking graphs.

Property	Money-laundering graph	NetFlow intrusion graph
Node meaning	Bank account	(port-bucket, protocol) proxy host
Edge meaning	Transaction	Network flow
Nodes	515,088	744
Edges	5,078,345	about 2,390,275
Edge features	amounts, currencies, format, time delta	43 NetFlow statistics
Positive rate	0.10 percent (laundering)	3.78 percent (attack)
Split	Time-based 70/15/15	Stratified random 70/15/15

4. Methodology

4.1 Edge-centric GNN backbone

The backbone is an E-GraphSAGE-style network (as in [6]) based on the inductive GraphSAGE aggregation scheme [2]. At every layer, a node receives messages from sampled neighbors and updates its representation (notably, each message comes from passing the associated edge feature vector through a small MLP followed by fusing that with the neighbor state). This way edge attributes fund the illustration, which is what a bad faith or assault alert generates. The concept here is that after the last layer, an edge representation based on its endpoint representations and the edge feature embedding need to be combined before passing this information into a classifier head that will output the illicit probability. The architecture is the same for both graphs except they differ on input feature dimension. Our hidden dimension is 128, number of propagation layers is 3, dropout set to 0.2 and edge feature mlp with hidden width 64.

Concretely, for layers $l = 1$ to L , let h denote a node embedding, φ the edge feature multilayer perceptron, σ a nonlinearity and the plus-in-circle symbol concatenation. Each layer computes a per-edge message, aggregates messages over sampled neighbors $N(v)$, and updates the node state, after which an edge representation is formed and scored:

$$m_{uv}^{(l)} = h_u^{(l-1)} \oplus \varphi(x_e) \quad (2)$$

$$a_v^{(l)} = \text{mean}_{u \in N(v)} m_{uv}^{(l)} \quad (3)$$

$$h_v^{(l)} = \sigma(W^{(l)} [h_v^{(l-1)} \oplus a_v^{(l)}]) \quad (4)$$

$$z_e = h_u^{(l)} \oplus h_v^{(l)} \oplus \varphi(x_e), \hat{y}_e = \sigma(w^T z_e + b) \quad (5)$$

Equation (2) is the point where edge features enter aggregation, and the ablation shows that removing this fusion (the structure-only baselines) is what causes precision-recall performance to collapse.

4.2 Class imbalance and losses

For both tasks, the tasks are imbalanced, with the money-laundering task extreme (1 per 1000 edges). We study two remedies. Focal loss de-emphasizes easy negatives and focuses training attention on hard, rare positives [13] and class-weighted binary cross-entropy uses the inverse class frequency to scale up the positive term. (We will see later how this choice interacts with how a deployment reads the model: focal loss maximizes ranking quality but can leave the default decision threshold predicting almost no positives, while weighted cross-entropy gives a usable operating point at the cost of some ranking quality. Here p is the predicted illicit probability, and the true-class probability equals p for positive edges and $1 - p$ for negative edges. With focusing parameter γ and balance parameter α , and with the positive weight set to the negative-to-positive ratio, the two losses are:

$$L_{\text{focal}} = -\alpha (1 - p_t)^{\gamma} \log(p_t) \quad (6)$$

$$L_{\text{wbce}} = -[w_{\text{pos}} \cdot y \log(p) + (1 - y) \log(1 - p)], \quad w_{\text{pos}} = N_{\text{neg}} / N_{\text{pos}} \quad (7)$$

4.3 Memory-efficient training

Training takes into account a single GPU and a 16 GB stem-memory constraint. Edges and features are read in batches, downcast to compact numeric types, the graph is trained with neighbor sampling (fan-outs of 15, 10 and 5 across the three layers) in mini-batches of 2,048 instead of materializing on the entire graph at once, and mixed precision arithmetic slashes memory utilization even further. Optimization uses Adam, a learning rate of 0.001, weight decay of 1e-5, gradient-norm clipping with a cosine schedule and early stopping on validation AUPRC. All configurations run across three seeds (42, 123 and 2024).

4.4 Baselines

Two baseline families provide context. Tabular models which only consider edge features but ignore graph structure (logistic regression, random forest, XGBoost and a multilayer perceptron) Structure-only GNNs (a plain graph convolutional network and a plain GraphSAGE) access connectivity but do not fuse edge features into aggregation. The comparison to those two family and the proposed model distinguishes the contribution of edge-feature fusion.

4.5 Explainability and risk scoring

We generate two distinct explanations for every case flagged. Each edge feature is responsible for the prediction using Integrated Gradients [17] and an edge-mask method similar to GNNExplainer [16] is used to highlight the local subgraph that contributed the most to a given prediction. Next, we aggregate the edge probabilities into a node-level risk score by summing the probabilities incident to each node and map scores to three tiers, Low, Medium and High, using fixed thresholds. It provides analyst with a case level reason and a worklist of the riskiest entities in order. Denote by $E(v)$ the set of edges incident to node v , the maximum incident edge probability is taken as risk score (a node is flagged at earliest interaction that appears suspicious); scores are tiered with respect to fixed thresholds:

$$r_v = \max_{e \in E(v)} \hat{y}_e, \quad \text{tier}(v) = \text{Low if } r_v \leq 0.33, \text{ Medium if } 0.33 < r_v \leq 0.66, \text{ High otherwise} \quad (8)$$

4.6 Evaluation metrics

Given this level of imbalance, accuracy is not useful, so we report the appropriate metrics here: AUROC (area under the receiver operating characteristic curve), AUPRC (area under the precision-recall curve), F1 score, precision, recall, Matthews correlation coefficient (MCC) and precision at k which measures how many in a set of top-ranked alerts are actually illicit. AUPRC and precision at k are the main metrics, as they represent how a team of analysts who have very limited capacity actually reads alerts. Sort edges according to descending score and write the label of the i -th ranked edge, and TP, TN, FP and FN the counts in confusion-matrix two main measures are:

$$P@k = (1/k) \sum_{i=1}^k y_{(i)} \quad (9)$$

$$\text{MCC} = (TP \cdot TN - FP \cdot FN) / \sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)} \quad (10)$$

5. Results

We first describe the class distributions, then the main comparison, then the ablation, and finally the explanation and risk analyses. All figures use identical settings across models and datasets. Reported numbers are test-set means over three seeds.

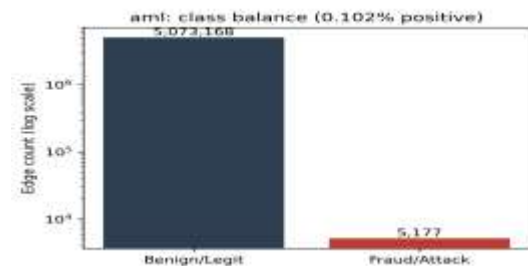


Figure 1a. Class balance of the money-laundering graph. Laundering edges are about 0.10 percent of all transactions.

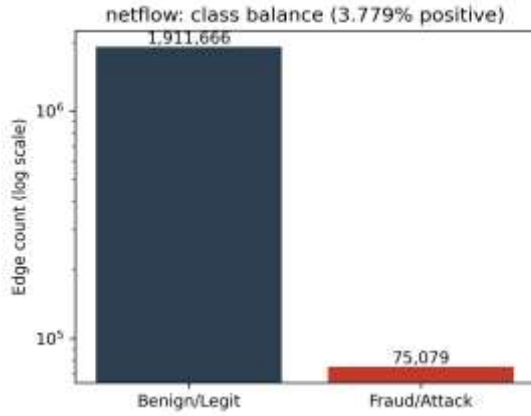


Figure 1b. Class balance of the NetFlow intrusion graph. Attack flows are about 3.78 percent of all flows.

5.1 Main comparison

The results for money laundering are shown in Table 2, and those for the intrusion attack in Table 3. Two findings stand out. First, edge-feature fusion is decisive. The two tasks collapse on GNNs that use only structure: for example the plain graph convolutional network reaches an AUPRC of just 0.004 on the money-laundering graph (the proposed edge-centric model gets to 0.166, a more than forty-fold improvement), and for the intrusion graph plain GraphSAGE reaches 0.498 while reaching 0.985 with our proposed model. The second is the intrusion task, where the proposed model performs best in absolute terms: it achieves overall highest AUPRC (0.985) and by considerable margin highest precision (0.958) of any method—i.e., fewer false alarms for same detections which will be operationally valuable.

We also report transparently and where interestingly the proposed model does not lead. On the very imbalanced money-laundering task, well-tuned tabular models remain competitive: multilayer perceptrons (MLP) and XGBoost AUPRCs of 0.210 and 0.172 respectively lead by a slim margin over our proposed model at 0.166, and the best precision@k respectively go to XGBoost again, which we attribute to the difficulty of learning from one positive per thousand edges on a proxy-free transaction graph and return to it in discussions. What is good about the graph model here is it doesn't give a higher number on headline but gives relational explanation and risk propagation tabular models could never compete with, and a single architecture that wins on the network surface as well.

Table 2. Money-laundering graph, test-set results (mean over three seeds). Best value per column in bold.

Model	AUROC	AUPRC	F1	Prec.	Recall	MCC	P@k
Logistic regression	0.938	0.025	0.027	0.014	0.931	0.104	0.000
Random forest	0.972	0.153	0.043	0.022	0.917	0.136	0.209
XGBoost	0.971	0.172	0.038	0.020	0.912	0.126	0.301
Multilayer perceptron	0.975	0.210	0.297	0.289	0.341	0.304	0.291
GCN (structure only)	0.569	0.004	0.000	0.000	0.000	0.000	0.011
GraphSAGE (structure only)	0.509	0.005	0.000	0.000	0.000	0.000	0.021

Proposed edge-centric GNN	0.974	0.166	0.000	0.000	0.000	0.000	0.248
---------------------------	-------	-------	-------	-------	-------	-------	-------

Note. Under focal loss the proposed model ranks illicit edges well (AUPRC 0.166, precision at k 0.248) but, at the default 0.5 threshold, predicts almost no positives, so its F1, precision and recall read as zero. The ablation shows that weighted cross-entropy produces a usable operating point on this task.

Table 3. NetFlow intrusion graph, test-set results (mean over three seeds). Best value per column in bold.

Model	AUROC	AUPRC	F1	Prec.	Recall	MCC	P@k
Logistic regression	0.996	0.881	0.891	0.806	0.996	0.892	0.840
Random forest	1.000	0.985	0.939	0.886	1.000	0.939	0.952
XGBoost	1.000	0.985	0.941	0.888	1.000	0.940	0.956
Multilayer perceptron	0.999	0.976	0.938	0.884	1.000	0.938	0.938
GCN (structure only)	0.585	0.099	0.073	0.897	0.038	0.181	0.096
GraphSAGE (structure only)	0.889	0.498	0.406	0.927	0.260	0.483	0.445
Proposed edge-centric GNN	0.999	0.985	0.938	0.958	0.919	0.936	0.946

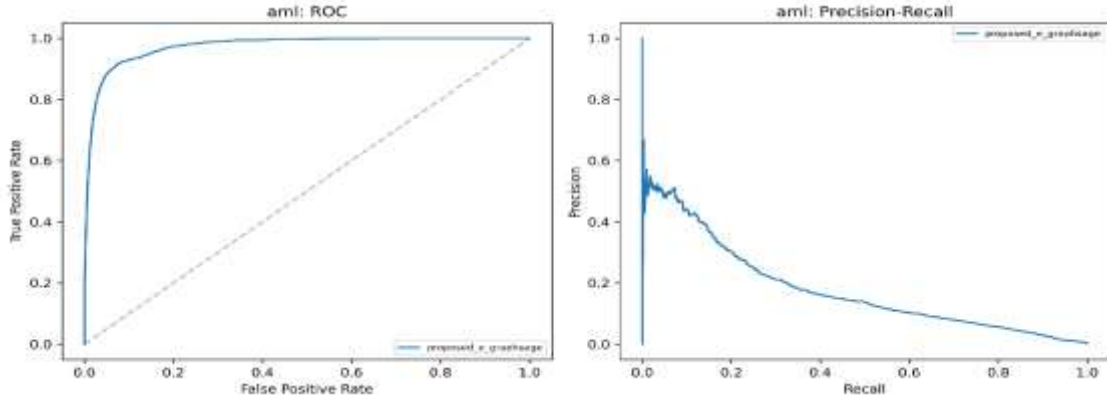


Figure 2. Receiver operating characteristic and precision-recall curves on the money-laundering graph.

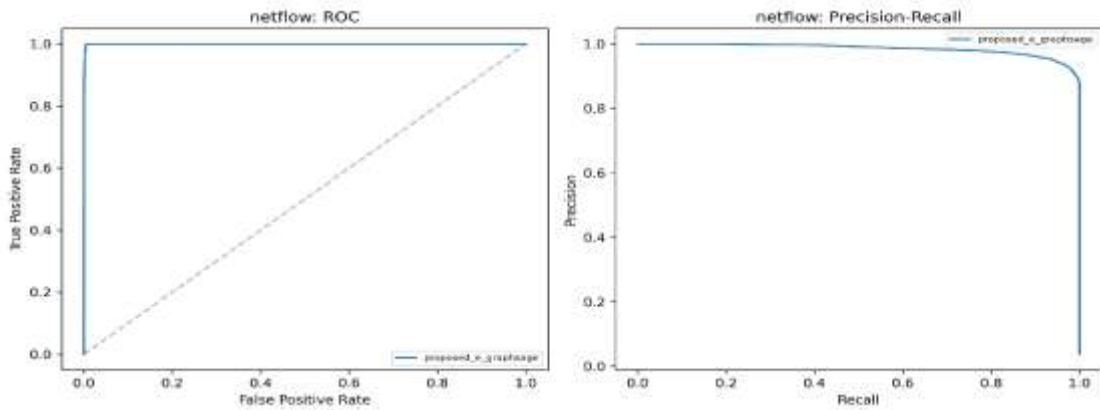


Figure 3. Receiver operating characteristic and precision-recall curves on the NetFlow intrusion graph.

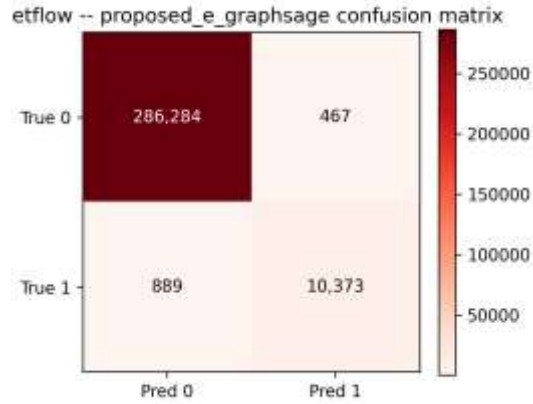


Figure 4. Confusion matrix of the proposed model on the NetFlow intrusion test set.

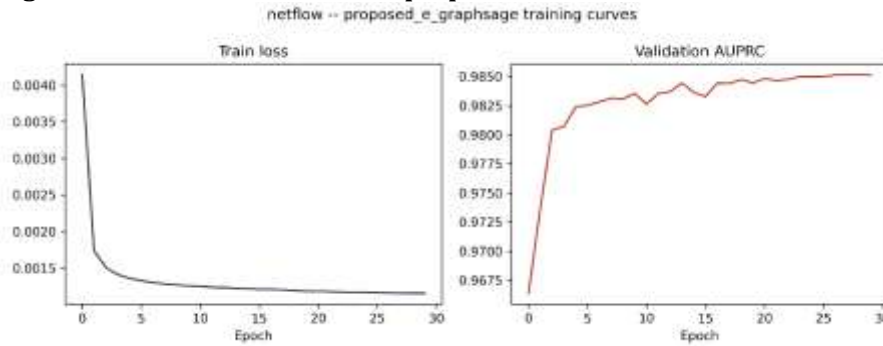


Figure 5. Training and validation curves of the proposed model on the NetFlow intrusion graph.

5.2 Ablation

In Table 4, we vary the loss function and number of propagation layers, with edge features fused at every layer. Both losses are strong on the intrusion graph, and depth has little influence indicating a sound configuration. The loss choice is material on the money-laundering graph. Focal loss achieves the best ranking (AUPRC 0.177 and max precision at k) but results in no true positives at the default threshold, while weighted cross-entropy trades some ranking for a deployable operating point with non-zero F1 near 0.16. This means reporting the ranking view (AUPRC, precision at k) and decision view (F1 at fixed threshold) together for extremely imbalanced financial data and calibrating the threshold itself according to analyst capacity instead of fixing it at 0.5.

Table 4. Ablation over loss and depth (edge features on). Test-set values.

Dataset	Loss	Layers	AUROC	AUPRC	F1	P@k
AML	focal	2	0.976	0.176	0.000	0.257
AML	focal	3	0.970	0.110	0.000	0.185
AML	focal	4	0.975	0.177	0.000	0.252
AML	weighted BCE	2	0.943	0.134	0.166	0.174
AML	weighted BCE	3	0.936	0.151	0.159	0.191

AML	weighted BCE	4	0.933	0.137	0.168	0.170
NetFlow	focal	2	0.999	0.984	0.935	0.943
NetFlow	focal	3	0.999	0.984	0.936	0.943
NetFlow	focal	4	0.999	0.984	0.938	0.943
NetFlow	weighted BCE	2	0.999	0.980	0.937	0.944
NetFlow	weighted BCE	3	0.999	0.980	0.936	0.945
NetFlow	weighted BCE	4	0.999	0.979	0.936	0.945

5.3 Explainability

An Integrated Gradients attribution for a flagged money-laundering case is shown in Figure 6. The payment format (an ACH transfer) and the time delta since the account was last transacted on are the strongest drivers while the amount paid has a positive effect, one currency indicator negatively. These are representative signals that an investigator could actually act on, and hence the transparency required/characteristic of a regulated environment. This edge-mask subgraph showing a subgraph of the explanation of an intrusion case, it has localized the flows and neighboring nodes that are mainly responsible for triggering this alert (Figure 7).

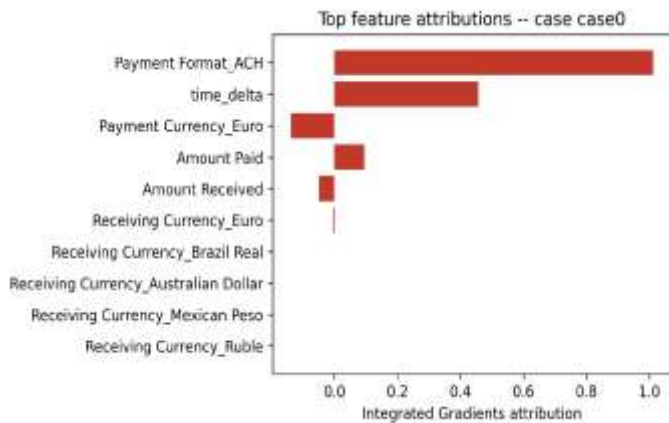


Figure 6. Integrated Gradients feature attribution for a flagged money-laundering edge.

Top 15 explanatory edges -- (case netflow_case0) (red = flagged edge endpoints)



Figure 7. Edge-mask subgraph explanation for a flagged intrusion edge.

5.4 Risk analysis

Ranked risk view by aggregating calibrated edge probabilities to nodes. Distributions for the Low, Medium and High risk tiers in the money-laundering graph appear in Figure 8, whilst Figure 9 visualises the highest-risk subgraph with density-based clustering of High-risk accounts into a handful of dense components. It is this node-level, tiered output that makes a per-edge classifier into an analyst worklist, and it is a capability lacking from the tabular baselines which have no notion of connected entities.

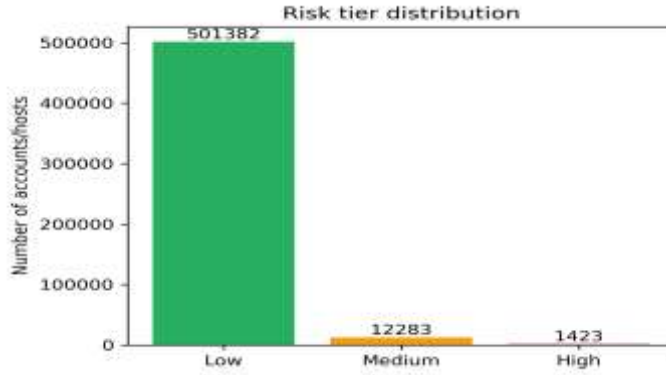


Figure 8. Distribution of node risk tiers on the money-laundering graph.

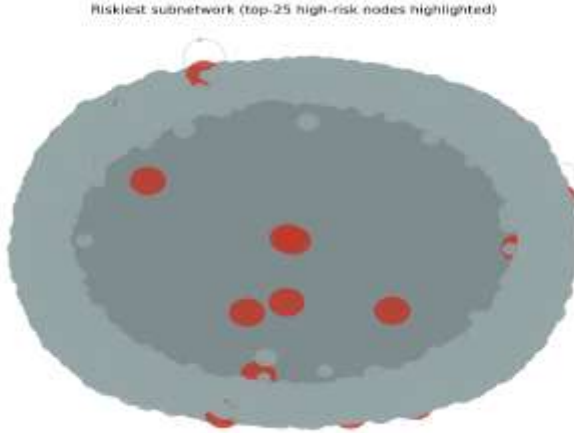


Figure 9. Highest-risk subgraph on the money-laundering graph. High-risk accounts concentrate in a few dense components.

6. Discussion

This result admits an unambiguous, circumscribed claim. The baseline is simple: a single edge-centric GNN in which edges are augmented with fused features into message passing, which constitutes an effective and widely transferable detector across both banking threat surfaces; edge feature fusion is necessary for anything to work as graph learning at all here: without it, the same connectivity yields fluctuating random precision-recall. The data is not publicly available. Overall, through top AUPRC and the lowest precision, which directly lightens analyst load, the proposed model is best on network surface.

The money-laundering aspect is more subtle. Except on AUPRC, where the tabular model strongly outperforms the graph (1 positive edge per thousand). We do not obscure this. The honest interpretation is that, for this synthetic transaction graph, the relational signal to which the model has access still cannot be competitive against feature-based learners with hand-tuned features on raw

ranking, even though it completely dominates structure-only graph baselines. And the graph model is still earning its place with capabilities that tabular models do not possess: Local relational explanations, node-level risk propagation, and one architecture to rule them all (also on the intrusion surface). That is a fair trade for the bank looking for one explainable relational system across both surfaces. The findings are qualified by several limitations that also explore the next steps for future work. The intrusion graph leverages a (port-bucket, protocol) proxy as host identity due to the unavailability of IP columns in the source data; thus, the network results reflect cross-domain transfer of the architecture rather than true host-level attribution; obtaining an IP-preserving NetFlow source would allow for a true host-topology evaluation. This data on money-laundering is entirely simulated, and absolute numbers can only be interpreted as relative comparisons, not estimates of production. The model we proposed is trained primarily with focal loss, which prefers ranking to a constant-threshold operating point; calibrated thresholds and cost-sensitive decision layers require independent study. Lastly, the use of a single dataset per surface; evaluation on more transaction and flow datasets would help better assess generalization.

7. Conclusion

This paper framed financial fraud and cyber-intrusion detection in banking as a single task, edge classification on a directed multigraph, and solved both with one edge-centric graph neural network. Across a large account-transaction graph and a NetFlow intrusion graph, edge-feature fusion proved decisive, lifting precision-recall performance far above structure-only graph baselines, and the proposed model achieved the best AUPRC and precision on the intrusion surface. On the extremely imbalanced money-laundering surface we reported, transparently, that strong tabular models remain competitive on raw ranking, while the graph model contributes relational explanations and node-level risk tiers that feature-based methods cannot. Together these results argue for a unified, explainable, relational approach to banking threat detection, and they identify IP-preserving flow data, calibrated decision layers and broader datasets as the next steps.

References

- [1] T. N. Kipf and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks. International Conference on Learning Representations (ICLR), 2017.
- [2] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive Representation Learning on Large Graphs. Advances in Neural Information Processing Systems (NeurIPS), 2017.
- [3] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph Attention Networks. International Conference on Learning Representations (ICLR), 2018.
- [4] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How Powerful are Graph Neural Networks? International Conference on Learning Representations (ICLR), 2019.
- [5] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A Comprehensive Survey on Graph Neural Networks. IEEE Transactions on Neural Networks and Learning Systems, 32(1):4-24, 2021.
- [6] W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, and M. Portmann. E-GraphSAGE: A Graph Neural Network based Intrusion Detection System for IoT. IEEE/IFIP Network Operations and Management Symposium (NOMS), 2022.
- [7] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann. NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems. Big Data Technologies and Applications (BDTA/WiCON), Springer, 2021.
- [8] N. Moustafa and J. Slay. UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems. Military Communications and Information Systems Conference (MilCIS), 2015.
- [9] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu. Enhancing Graph Neural Network-based Fraud Detectors against Camouflaged Fraudsters. ACM International Conference on Information and Knowledge Management (CIKM), 2020.
- [10] Y. Liu, X. Ao, Z. Qin, J. Chi, J. Feng, H. Yang, and Q. He. Pick and Choose: A GNN-based Imbalanced Learning Approach for Fraud Detection. The Web Conference (WWW), pp. 3168-3177, 2021.

- [11] M. Weber, G. Domeniconi, J. Chen, D. K. I. Weidele, C. Bellei, T. Robinson, and C. E. Leiserson. Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics. arXiv:1908.02591, 2019.
- [12] M. Weber, J. Chen, T. Suzumura, A. Pareja, T. Ma, H. Kanezashi, T. Kaler, C. E. Leiserson, and T. B. Schardl. Scalable Graph Learning for Anti-Money Laundering: A First Look. arXiv:1812.00076, 2018.
- [13] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal Loss for Dense Object Detection. IEEE International Conference on Computer Vision (ICCV), 2017.
- [14] T. Chen and C. Guestrin. XGBoost: A Scalable Tree Boosting System. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 785-794, 2016.
- [15] L. Breiman. Random Forests. Machine Learning, 45(1):5-32, 2001.
- [16] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. GNNExplainer: Generating Explanations for Graph Neural Networks. Advances in Neural Information Processing Systems (NeurIPS), 2019.
- [17] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic Attribution for Deep Networks. International Conference on Machine Learning (ICML), 2017.
- [18] E. A. Lopez-Rojas, A. Elmir, and S. Axelsson. PaySim: A Financial Mobile Money Simulator for Fraud Detection. European Modeling and Simulation Symposium (EMSS), pp. 249-255, 2016.
- [19] Z. Liu, Y. Dou, P. S. Yu, Y. Deng, and H. Peng. Alleviating the Inconsistency Problem of Applying Graph Neural Network to Fraud Detection. ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1569-1572, 2020.