# Detecting Financial Fraud Using Graph Neural Networks


Università della Svizzera italiana | Faculty of Informatics | Institute of Computing CI

Julien Schmidt, Dimosthenis Pasadakis, Madan Sathe*, Olaf Schenk

*Ernst and Young (EY)



Weber, Mark, Jie Chen, Toyotaro Suzumura, Aldo Pareja, Tengfei Ma, Hiroki Kanezashi, Tim Kaler, Charles E. Leiserson, and Tao B. Schardl. "Scalable Graph Learning for Anti-Money Laundering: A First Look." arXiv, November 30, 2018. https://doi.org/10.48550/arXiv.1812.00076

Elliott, Andrew, Mihai Cucuringu, Milton Martinez Luaces, Paul Reidy, and Gesine Reinert. "Anomaly Detection in Networks with Application to Financial Transaction Networks." arXiv, May 24, 2019. https://doi.org/10.48550/arXiv.1901.00402.

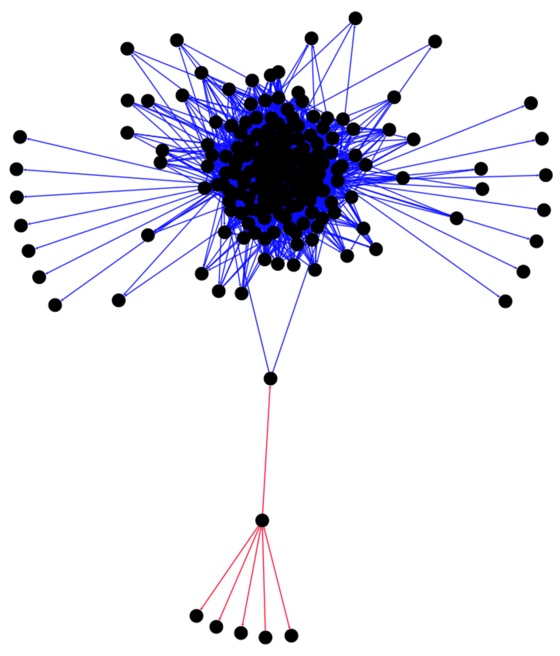

## Problem Description

In this study we tackle the problem of money laundering detection in large-scale financial networks. We generate synthetic graph-structured data emulating a financial system with embedded money laundering topologies. We employ various Graph Deep Learning techniques and compare their effectiveness in detecting fraudulent accounts.

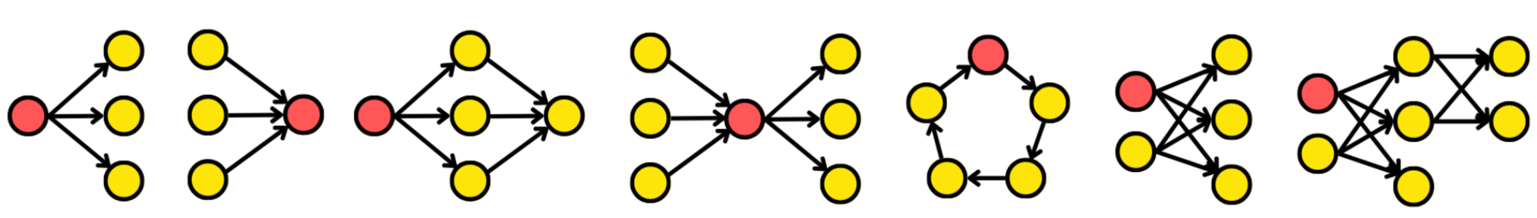
## Synthetic Dataset Generation

Graph $G(V,E)$ directed-multigraph with $n$ vertices and $m$ edges.

- Vertices: Bank accounts
- Edges: Transactions between accounts
- Financial graph-structured dataset generated using AMLsim. We specify number of normal and anomalous accounts, types of money laundering topologies, and the duration of simulation.
- Post-processing of generated dataset. Using edge-level features (transaction amount as weights of the edges), and node connectivity metrics, we compute node-level features used for training GNN models.



## Money Laundering Topologies



## Feature Generation

A set of node-level features are generated for each account.

- **GAW**: geometric average of weights. For weight $w_j^i$, of neighbors of node $i$ of total degree $d_t(i)$.

$$GAW(i) = \left(\prod_{j=1}^{d_t(i)} w_j^i\right)^{d_t(i)^{-1}}$$

- **GAW10** and **GAW20**: The GAW of each node computed for the largest 10% and 20% of its connected edge weights respectively.
- **Standardized Node Degree**: Where $\mathbf{d_t}$ is a vector of all node total degrees in graph G.

$$SND(i) = \frac{d_t(i) - \text{mean}(\mathbf{d_t})}{\text{std}(\mathbf{d_t})}$$

- **Node Degree (In/Out)** and **Unique Node Degree (In/Out)** when Graph G is reduced to Digraph and only one edge between nodes is permitted.
- **Degree Frequency**: Node's Total Unique Degree divided by Nodes Total Degree.

$$d_f(i) = \frac{d_{ut}(i)}{d_t(i)}$$

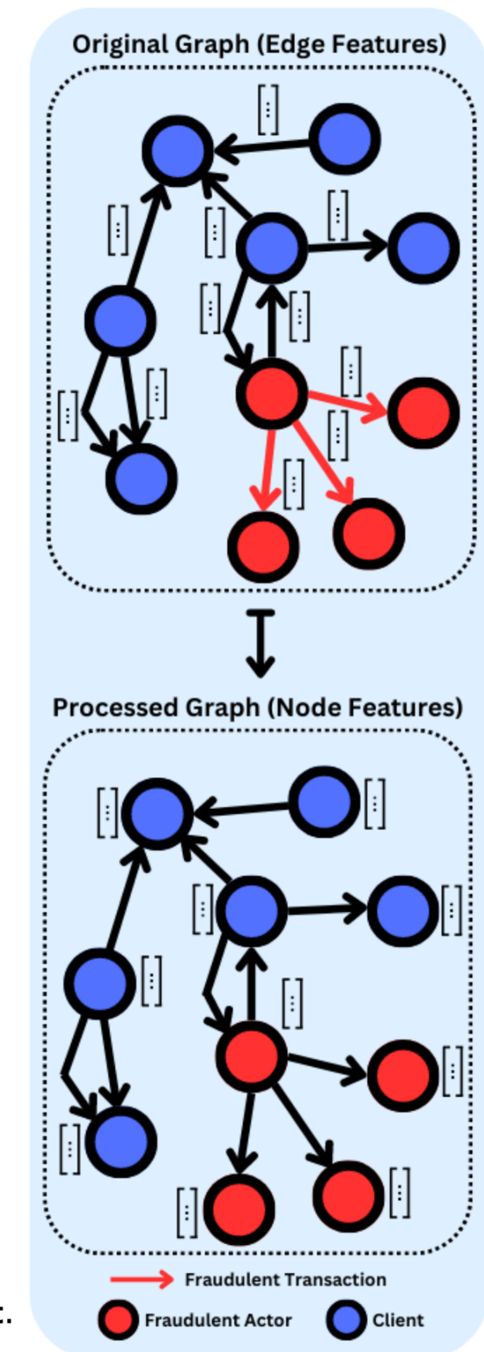
- **Node's Community Edge Density**: Communities $C(i)$ of a node $i$ are computed using the Louvain Algorithm [Blondel et al., 2008]. Nodes are given their respective communities' edge density.

$$ED(i) = \frac{2 \cdot |E_{C(i)}|}{|V_{C(i)}| \cdot (|V_{C(i)}| - 1)}$$

- **Scaled Node's Community Edge Density** divided by Community Size.

$$SED(i) = \frac{ED(i)}{|V_{C(i)}|}$$

- A node's **(min/max/mean/std./total) transaction amount in**.
- A node's **(min/max/mean/std./total) transaction amount out**.




## Graph Neural Network Models And Datasets

**Message Passing Neural Network (MPNN)** [J. Gilmer et al., 2017]

$$h_i^{l+1} = \tanh\left(h_i^l \Theta_1^l + \sum_{j \in N(i)} \widetilde{a}_{ji} \cdot h_j^l \Theta_2^l\right)$$

Where $h_i^l$ is the $l^{\text{th}}$ convolutional layer for node $i$, $\Theta_1^l$ and $\Theta_2^l$ are learnable parameter matrices, $\widetilde{a}_{ji}$ is an element of the graph shift operator for nodes $j$ and $i$, and $\sum_{j \in N(i)}$ is the aggregator function for neighborhood of node $i$.

**GraphSAGE** [Hamilton et al., 2017]

**Graph Convolutional Network (GCN)** [Kipf and Welling, 2017]

**Graph Attention Network (GAT)** [Veličković et al., 2018]

**Graph Isomorphism Network (GIN)** [Xu et al., 2019]

**Datasets**

Four datasets generated with varying number of anomalous nodes $|V_A|$, number of normal nodes $|V_N|$, and ratios of anomalous to normal nodes. Number of nodes remains constant across all datasets. Only the number of anomalous accounts and number of edges changes.

Datasets

| Dataset | Balance (Anom./Normal) | $\|V\|$ | $\|V_A\|$ | $\|V_N\|$ | $\|E\|$ |
|---|---|---|---|---|---|
| 1 | 55%/45% | 60,215 | 32,877 | 27,338 | 1,076,063 |
| 2 | 11%/89% | 60,215 | 6,581 | 53,634 | 1,001,080 |
| 3 | 5%/95% | 60,215 | 3,279 | 56,936 | 992,675 |
| 4 | 2%/98% | 60,215 | 1,288 | 58,927 | 986,789 |

## Data Processing and Model Training Pipeline

**Algorithm 1** Node and Edge Feature Generation

**Input:** $G(V,E)$ ▷ generated dataset from AMLsim
**Output:** $A_V, A_E, el, y$ ▷ node attributes, edge attributes, edge list, target

```
A_V ← empty(|V|, 1)
A_V.append(BasicNodeTests(G(V, E)))              ▷ GAW, Std. Degree
A_V.append(CommunityDetection(G(V, E)))          ▷ Louvain Method
A_V.append(TransactionStatistics(G(V, E)))       ▷ total amount in, ect..
A_E ← E.weights                                  ▷ transfer amount
el ← E.edge_list                                 ▷ [source, target] of shape (|E| × 2)
y ← node class label                             ▷ boolean {0, 1}
return A_V, A_E, el, y
```

**Algorithm 2** GNN Model Training Pipeline

**Input:** $A_V, A_E, el, y$
**Output:** model

```
k ← 2                                            ▷ number of neighborhood layers
hidden_size ← A_V.num_features                   ▷ number of node features
model = GNNModel(hidden_size, k)
train, valid, test ← random_split(A_V, A_E, el, y)   ▷ random indices splits
for epoch do
    model.train(A_V, A_E, el, y, train)
    model.valid(A_V, A_E, el, y, valid)
end for
return model
```

## Numerical Results

### Performance of GNN Models Across Datasets

Dataset 1
55% anomalous, 45% normal

| Model (Best Params) | Precision ↑ | Recall ↑ | F1-Score ↑ | AUC ↑ |
|---|---|---|---|---|
| MPNN, k=5, h=64 | .95 | **.95** | .95 | .94 |
| SAGE, k=3, h=64 | **.97** | **.95** | **.96** | **.95** |
| GCN, k=5, h=64 | .88 | .94 | .91 | .90 |
| GAT, k=3, h=32 | .90 | .92 | .91 | .89 |
| GIN, k=3, h=32 | .95 | .90 | .92 | .92 |

Dataset 2
11% anomalous, 89% normal

| Model | Precision ↑ | Recall ↑ | F1-Score ↑ | AUC ↑ |
|---|---|---|---|---|
| MPNN, k=4, h=8 | .87 | **.80** | .84 | **.89** |
| SAGE, k=5, h=64 | **.97** | .71 | **.86** | .88 |
| GCN, k=3, h=32 | .83 | .69 | .75 | .84 |
| GAT, k=3, h=16 | .93 | .65 | .77 | .82 |
| GIN, k=2, h=32 | .98 | .76 | **.86** | .88 |

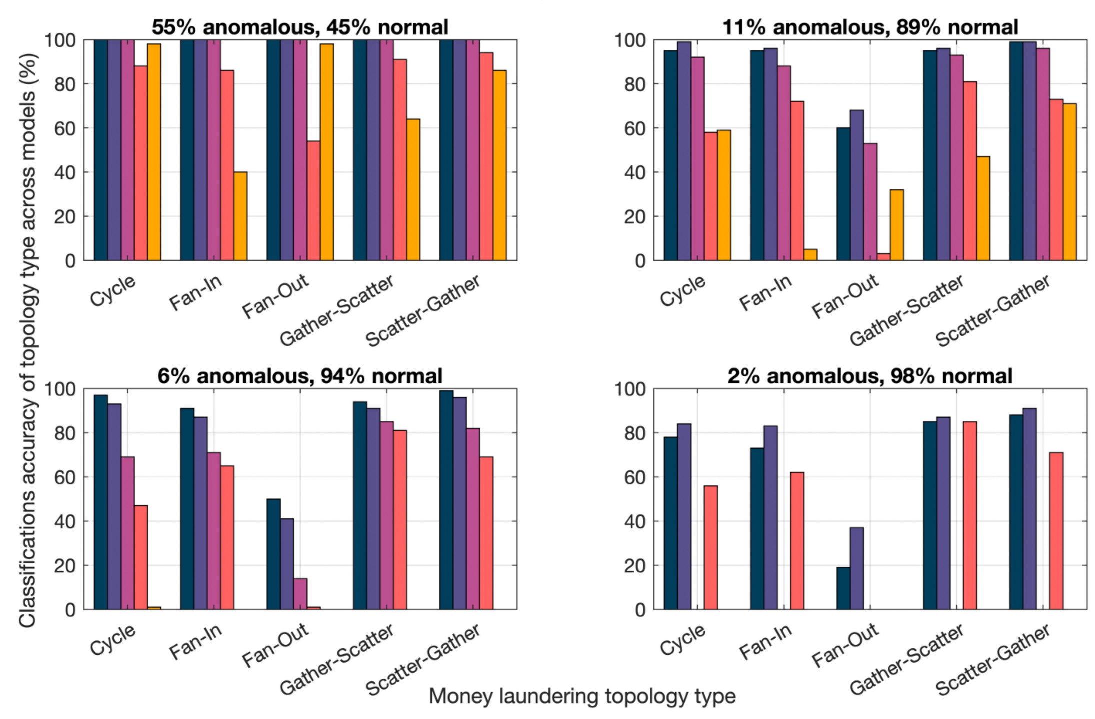
Dataset 3
5% anomalous, 95% normal

| Model | Precision ↑ | Recall ↑ | F1-Score ↑ | AUC ↑ |
|---|---|---|---|---|
| MPNN, k=5, h=64 | .93 | .75 | .83 | .87 |
| SAGE, k=6, h=8 | .97 | **.81** | **.88** | **.90** |
| GCN, k=2, h=8 | .85 | .68 | .76 | .84 |
| GAT, k=3, h=16 | .94 | .65 | .77 | .82 |
| GIN, k=2, h=16 | **.98** | .72 | .83 | .86 |

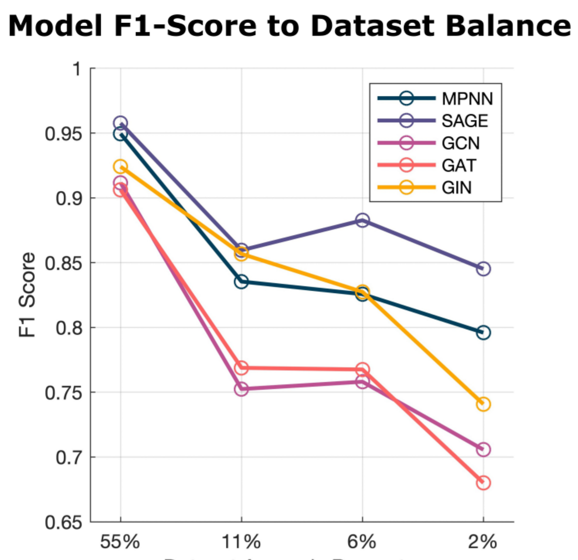
Dataset 4
2% anomalous, 98% normal

| Model | Precision ↑ | Recall ↑ | F1-Score ↑ | AUC ↑ |
|---|---|---|---|---|
| MPNN, k=5, h=64 | .92 | .70 | .80 | .85 |
| SAGE, k=6, h=8 | .95 | **.76** | **.85** | **.88** |
| GCN, k=2, h=8 | .86 | .60 | .71 | .80 |
| GAT, k=5, h=16 | .92 | .54 | .68 | .77 |
| GIN, k=2, h=32 | **.98** | .60 | .74 | .80 |

### Model Performances on Topology Types Across Dataset Balances

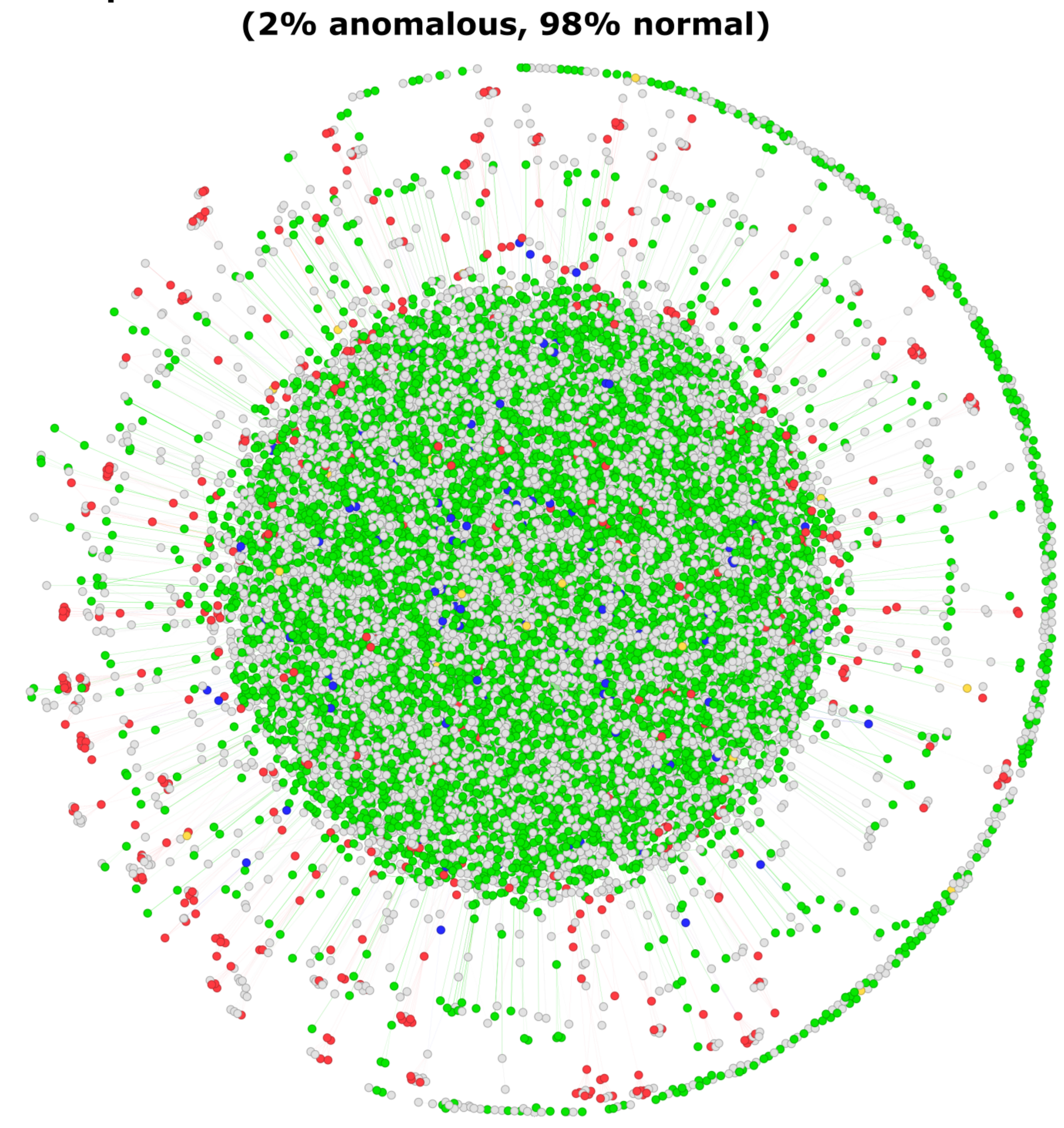



### Model F1-Score to Dataset Balance




### Model Training Times


### GraphSAGE Node Classification Results for Dataset 4 (2% anomalous, 98% normal)



Red: True Positives (2.31%)
Yellow: False Positives (0.23%)
Green: True Negatives (46.52%)
Blue: False Negatives (0.91%)
White: Training Set (50.03%)