Classifying Illicit Bitcoin Transactions

w/ Graph Neural Networks ⇲

PRESENTED BY:
Xuhui Zhan, Siyu Yang, Tianhao Qu

Content

Problem Statement

what

This project addresses the challenge of detecting illicit activity in cryptocurrency transaction networks by analyzing structural graph properties and applying Graph Neural Network (GNN) models to classify unlabeled nodes in a Bitcoin transaction graph.

why

Traditional classification approaches may achieve high accuracy but often fail to preserve critical structural properties of transaction networks, which are essential for realistic detection of illicit financial activity. A model that maintains these graph properties while achieving high accuracy is more likely to generalize well to real-world applications.

how

This project is divided into 3 parts:

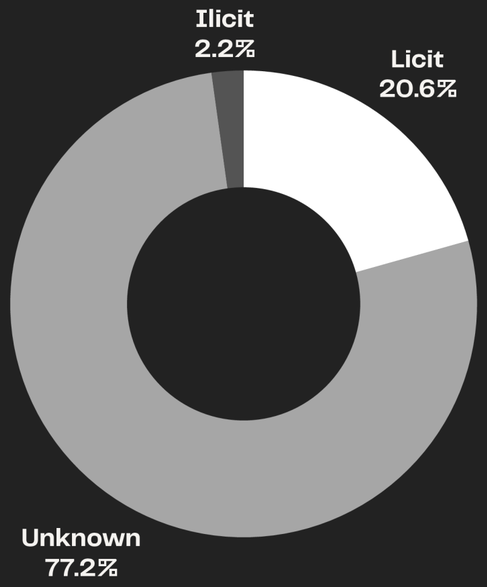
1. Direct classification with base models
2. Data Augmentation with base models and label propagation
3. Graph Property Preservation Analysis (Training (licit and illicit) Unknown (pred_licit and illicit))

About the Graph ↘

Graph Properties	Details
Source	Provided by Elliptic, a blockchain analytics firm; includes 203,769 Bitcoin transactions and 234,355 edges, transactions labeled as licit, illicit, or unknown.
Graph Type	Time-evolving transaction graph over 49 steps; each node is a transaction, and edges represent Bitcoin flows.
Features	166 features per node, including transaction-level statistics (e.g. time, amount) and aggregated metrics over time (e.g. degree, centrality).
Why this graph	Real-world, large-scale blockchain data with challenges like class imbalance, label sparsity, and evolving structure—ideal for testing GNNs in semi-supervised and temporal settings.

2.2% ILLICIT

Transactions that are confirmed to be linked to illegal activity.



20.6% LICIT

Legitimate transactions not associated with any criminal activity.

77.2% UNKNOWN

Dataset but haven't been labeled—their nature is unclear or undetermined.

Class Distribution

Data Preprocessing

Graph Construction & Cleaning

- Extract node features and edge lists from different csv files
- Mapped transaction IDs to indices to make data compatible with PyTorch Geometric.
- Separate the graph into different sets: Known (Train, Val, Test) and Unknown

Exploratory Analysis

- Degree Distributions: Both in-degree and out-degree follow power-law, typical in financial graphs.
- Community Detection: Used Louvain algorithm to identify transaction clusters—potential signs of coordinated behavior.
- Network Structure: One dominant connected component contains the majority of labeled transactions.

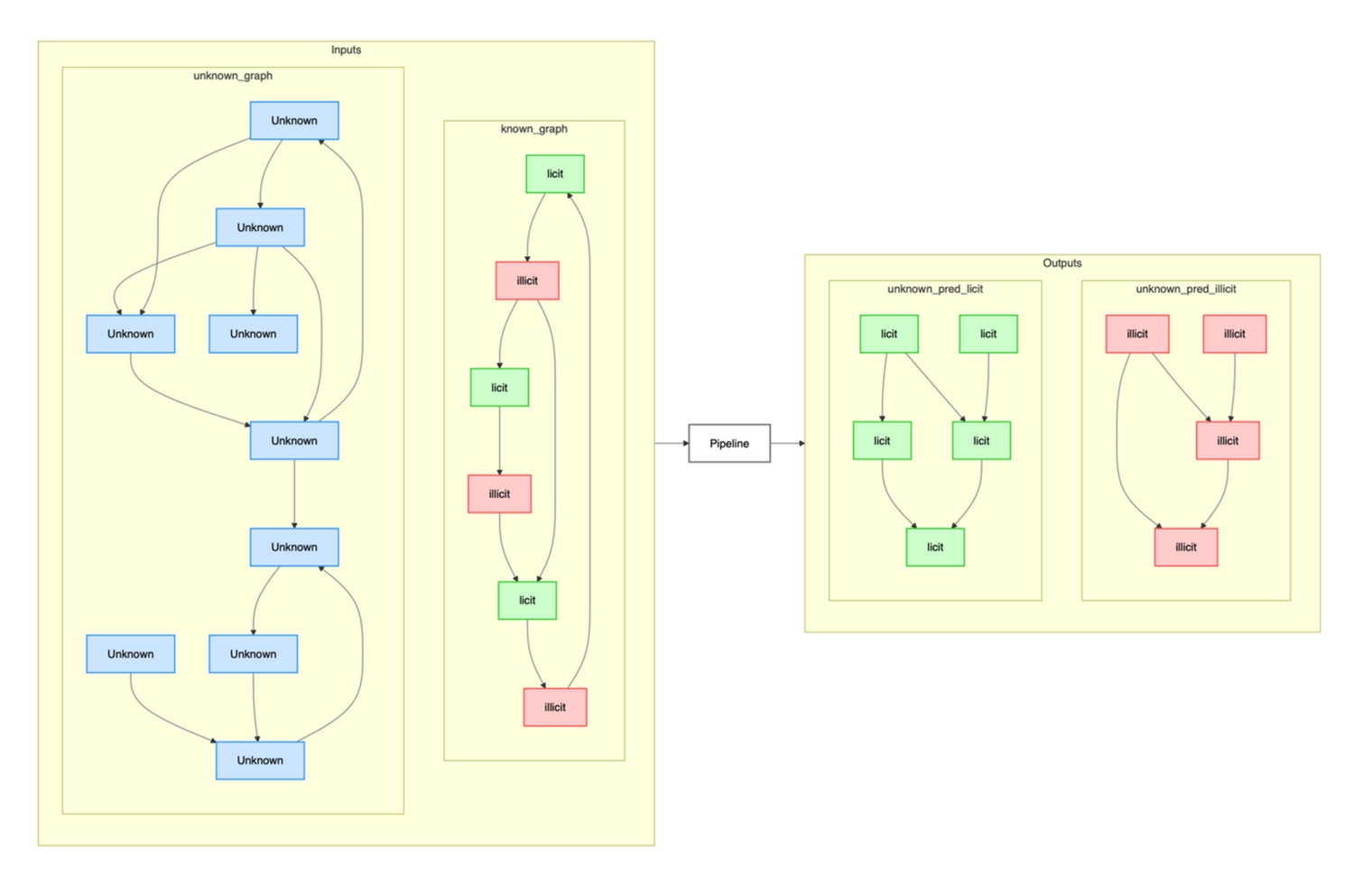
Methodology ↘

1. Without Data Augmentation
   - GraphSAGE: Trained on the original training set; predicts labels for all unknown nodes.
   - GAT: Same setup as above, using a Graph Attention Network.

2. With Data Augmentation (30% of unknown nodes added to training set)
Each of the following uses a different augmentation strategy to **label unknown nodes before retraining**:
   - **Label Propagation Augmentation**
     - GraphSAGE → trained on augmented data
     - GAT → trained on augmented data
   - **GraphSAGE-based Augmentation**
     - GraphSAGE → trained on GraphSAGE-augmented data
     - GAT → trained on GraphSAGE-augmented data
   - **GAT-based Augmentation**
     - GraphSAGE → trained on GAT-augmented data
     - GAT → trained on GAT-augmented data

Methodology ↘

Each pipeline generates different predictions for **Unknown nodes,** classifying them as either Licit or Illicit, e.g. unknown_pred_licit subgraph created by pipeline GAT_SAGE.



Methodology ↘

To evaluate these subgraphs:

1. **Compute Subgraph Properties**
   - For each subgraph, we compute graph structural metrics (e.g. Largest Component Ratio).
2. **Create Ideal Baseline**
   - We bootstrap samples from the original training set to estimate the natural variance of each subgraph metric.
   - These serve as our ideal variance baseline — what we expect from correct classifications.
3. **Compare with Unknown Predictions**
   - For each predicted subgraph (e.g., unknown_pred_licit), we:
     - Calculate the distance from the training subgraph metrics.
     - Normalize this distance using the bootstrapped variance.

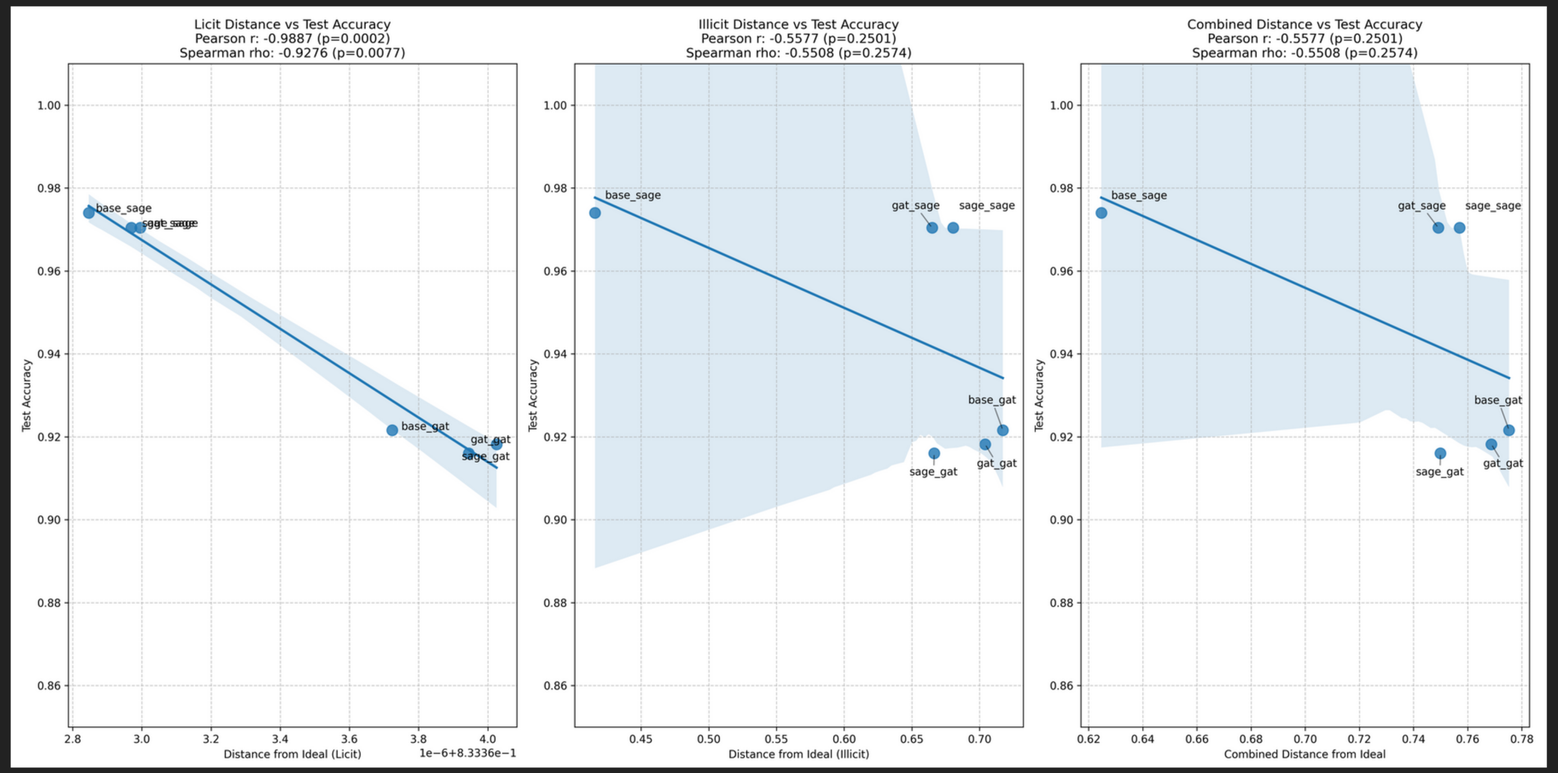
If the model predictions are perfectly accurate, the predicted licit/illicit subgraphs should resemble real-world ones (as those in training set) — and thus, have a small Distance to Ideal Baseline.

Results: Model Performance ↘

Table 1: Model Performance: Train, Val, Test

Model	Final Training Loss ↓	Validation Accuracy ↑	Test Accuracy ↑
base_gat	0.272	0.921	0.921
base_sage	0.090	**0.977**	**0.974**
label_propagation_gat	0.239	0.912	0.907
label_propagation_sage	0.110	0.964	0.965
gat_gat	0.182	0.906	0.918
gat_sage	**0.047**	0.974	0.970
sage_gat	0.188	0.910	0.916
sage_sage	0.049	0.973	0.970

Results ↘



Strong negative correlation for licit preservation (r=–0.99, p=0.0002)
Moderate correlation for illicit preservation (r=–0.56, p=0.25)

- Models that better preserve licit network structure achieve significantly higher test accuracy
- Base GraphSAGE's **neighborhood sampling** approach maintains critical transaction patterns without distortion
- **Structural preservation** appears to be a strong predictor of model reliability in cryptocurrency transaction networks

Conclusion ↘

1. Preserving authentic network topology is critical for reliable fraud detection.
2. The distance-to-ideal metric provides a complementary evaluation beyond traditional accuracy measures.
3. GraphSAGE's neighborhood sampling preserves essential graph properties more effectively than GAT's attention.
4. Preserving licit subgraph structure is more indicative of model reliability than preserving illicit patterns.

Future work includes extending to temporal networks, developing structure-optimized models, improving explainability, and scaling to larger cryptocurrency datasets.

Q&A

Github Repo: https://github.com/xuhuizhan5/BTCGraphGuard

Appendix: Structure Preservation Metrics

01

+ Six Critical Graph Properties

1. **Homophily**: Same-class connection tendency
2. **Density**: Actual-to-possible connections ratio
3. **Clustering Coefficient**: Community tightness
4. **Degree Centrality**: Node connection influence
5. **Betweenness Centrality**: Bridge node importance
6. **Largest Component Ratio**: Network fragmentation

02

+ Distance-to-Ideal Calculation

- Measure similarity between training and predicted subgraphs for each property
- Bootstrap sampling to establish natural variation baseline
- Normalize differences using bootstrap standard deviation
- Distance = |Normalized Similarity – 1.0|
- Perfect preservation = 0 distance, larger values = greater distortion

Appendix: Important formulas ↘

Similarity and Distance Metrics

$$\text{Raw Similarity}(v_1, v_2) = \begin{cases} 1.0 & \text{if } v_1 = 0 \text{ and } v_2 = 0, \\ 0.0 & \text{if } v_1 = 0 \text{ or } v_2 = 0, \\ 1 - \min\left(\dfrac{|v_1 - v_2|}{\max(|v_1|, |v_2|)}, 1\right) & \text{otherwise.} \end{cases}$$

$$\text{Bootstrap Normalized Similarity} = \begin{cases} 1.0 & \text{if } \sigma_{\text{bootstrap}} = 0 \text{ and } |v_{\text{pred}} - v_{\text{train}}| = 0, \\ 0.0 & \text{if } \sigma_{\text{bootstrap}} = 0 \text{ and } |v_{\text{pred}} - v_{\text{train}}| \neq 0, \\ \max\left(0,\, 1 - \min\left(\dfrac{|v_{\text{pred}} - v_{\text{train}}|}{\sigma_{\text{bootstrap}} \cdot 3}, 1\right)\right) & \text{otherwise.} \end{cases}$$

where

v_{pred} is the predicted value, v_{train} is the training value, $\sigma_{\text{bootstrap}}$ is the standard deviation from bootstrap samples.

$$\text{Distance from Ideal} = \left|\text{Normalized Similarity} - 1.0\right|.$$

$$\text{Combined Distance} = \frac{\text{Distance from Ideal}_{\text{licit}} + \text{Distance from Ideal}_{\text{illicit}}}{2}.$$

Appendix: Detailed Methodology↘

01

02

Experimental Design

- Base models: GAT vs. GraphSAGE (**2** models total)
- Data augmentation and then train with base models (**30%** unknown nodes):
  - Label propagation
  - Cross-model augmentation (e.g., GAT→SAGE, **6** models total)
- Research question: Does structure properties preservation (Original Training Set VS Unknown predict) correlate with performance?

+ A New Metric for Subgraph Analysis

- Extract properties from two subgraph types by each of the **8** models:
  - Training set: known licit and illicit nodes
  - Prediction set: unknown nodes classified
- Bootstrap process for establishing baseline:
  - Sample **70%** of training nodes randomly
  - Calculate subgraph property differences from full training set among **6** properties
  - Repeat **10** times to determine natural variance
  - Use variance to normalize prediction–training differences
  - Quantify proposed "**distance to ideal**" as normalized deviation from expected properties for measuring structure preservation (Details in later part)