G-FRAUD: Detecting Collusive E-commerce Fraud with Heterogeneous Graph Neural Networks

Om Joshi[a,*], Ritvik Koneru[a,*]

[a] *University of North Texas, Denton, Texas, United States of America*

Abstract

E-commerce sites and Buy Now, Pay Later (BNPL) services have grown exponentially over the past years, parallel with the corresponding growth of highly sophisticated fraud. Classical fraud detection systems, that inspect transactions individually, usually don't uncover collusive fraud rings that take advantage of relationships among multiple addresses and cards. This paper introduces a graph-based framework for detecting fraud that Model relationships between payment cards and shipping addresses as a heterogeneous network. By representing rich behaviors and numerical features into a graph and using a Graph Neural Network (GNN), this method is capable of discovering the relational structure that separates collusive fraud as opposed to legitimate behaviour. The introduced method focuses on learning connectivity patterns as opposed to alone anomalies, hence leaving a more effective tool for uncovering coordinated attacks. This work points to the capabilities that graph-based relational learning holds for fortifying fraud prevention systems used by e-commerce sites as well as other financial platforms.

Keywords: Fraud detection, Graph Neural Networks (GNNs), E-commerce security, Heterogeneous networks, Relational learning, Buy Now Pay Later (BNPL)

1. Introduction

1.1. Background

E-commerce growth, combined with the rise of convenient payment services, including Buy Now, Pay Later (BNPL) services, has changed the face of consumer financial conduct. As the services widen availability and ease, they've also opened up the floodgates to newer modes of fraud. Criminals are now more routinely active as organized rings, taking advantage of pay ecosystem vulnerabilities. As an instance, the fraudsters would ship merchandise ordered through various stolen credit cards to a limited group of addresses run as a control group. These networked practices are challenging to identify through traditional fraud detection methodologies as they are based on the independent assessment of transactions.

1.2. The Limits of Transactional Analysis

Classical machine learning fraud models, such as logistic regression, random forests, and boosted trees, are very effective at pointing out abnormal single transactions. They do not, however, have the capability to discover the higher-order relational patterns. An easy transaction-level anomaly would be safe looking individually but suspicious looking in the broader context over the card and address network. An example would be ten varying credit cards all of a sudden shipping to a newly added address, appearing highly suspicious, even as an individual transaction, looking like past legitimate activity. This relational aspect drives the need for graph-based methods that capture the relationships between entities [1][2]. Such methods leverage Graph Neural Networks (GNNs) to learn hidden dependencies that can expose complex fraudulent behavior [3][4].

1.3. Our Objective

The goal of this work is to build a graph-convolution-based fraud detection system that detects coordinated, collusive activities from e-commerce transaction data fraud. Instead of taking transactions as isolated records, payment cards and shipping addresses are represented as nodes in a heterogeneous graph. This provides a method where relationships among entities are modeled explicitly and used during the learning process. Utilizing Graph Neural Networks (GNNs) on this representation, the system is able to learn relational structure indicating fraud rings, complementing conventional detection techniques. Broadly, the intent is to show the potential of relational learning for fraud prevention as well as outline a framework that could be extended to large financial systems.

2. Methodology

2.1. Dataset and Pre-Processing

These experiments used the IEEE-CIS fraud detection dataset[5], composed of many anonymized e-commerce transactions that are either fraudulent or legitimate. To ready this dataset for graph-based modeling, numerous preprocessing steps were needed. Unique identifiers were first created for cards as well as addresses, making use of label encoding to be certain that the identifiers were integer-based and amenable

[*]These authors contributed equally to this work.

to graph construction. Transactions that were missing important attributes, like card or address details, were dropped to ensure structural consistency after creating the resulting graph. Due to the dataset's size, we used a stratified sample that retained the original distribution of fraud while keeping computations within sensible limits. Numerical features like transaction amount were then standardized to enhance the stability that is trainable, as were categorical variables like product category as well as the type of the device, which were one-hot encoded to gain additional behavioral insights.

2.2. *Heterogeneous Graph Construction*

It was modeled the dataset as a bipartite graph that comprised of two types of nodes, payment cards, and delivery addresses. The lines between the nodes indicated the transactions, with addition of reverse edges so that the graph contains bidirectional message passing the training period. Node features were preprocessed based on the summarization of the transaction-level information into the node-level embeddings. For the numerical features such as the transaction amount, statistics that encompassed the mean, standard deviation, max, as well as the count were utilised to summarise the value distribution corresponding to each node. In the case of the categorical variables such as the product code or the type of devices, the one-hot encoded vectors were added together to get the behavioral profiles. This guaranteed that both the node included both the statistical as the behavioral indicators which mirrored the transactional history.

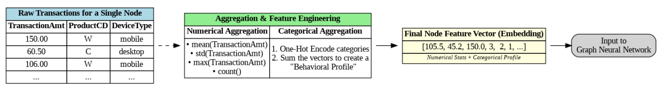
Raw Transactions for a Single Node
TransactionAmt | ProductCD | DeviceType
150.00 | W | mobile
60.50 | C | desktop
106.00 | W | mobile
... | ... | ...
Aggregation & Feature Engineering
Numerical Aggregation
• mean(TransactionAmt)
• std(TransactionAmt)
• max(TransactionAmt)
• count()
Categorical Aggregation
1. One-Hot Encode categories
2. Sum the vectors to create a "Behavioral Profile"
Final Node Feature Vector (Embedding)
[105.5, 45.2, 150.0, 3, 2, 1, ...]
Numerical Stats + Categorical Profile
Input to Graph Neural Network

Figure 1: Node Aggregation Flowchart

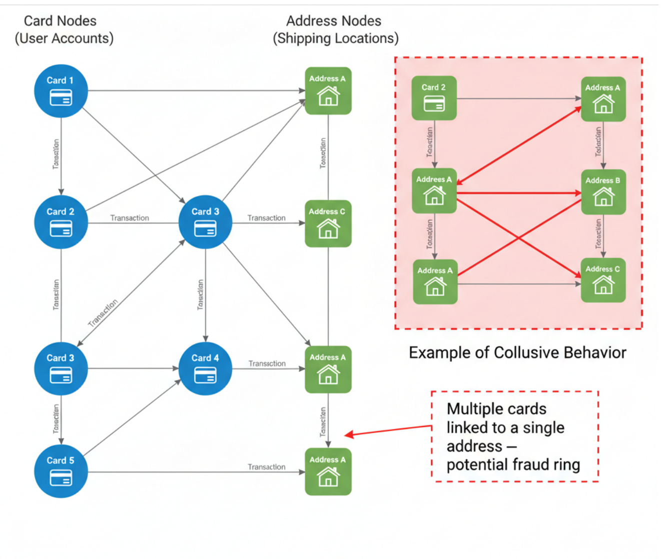
Card Nodes (User Accounts)
Address Nodes (Shipping Locations)
Card 1
Card 2
Card 3
Card 4
Card 5
Address A
Address C
Address B
Transaction
Example of Collusive Behavior
Multiple cards linked to a single address – potential fraud ring

Figure 2: Transactions as Edges in Bipartite Graph

2.3. *Model Architecture (SAGEConv GNN)*

The graph was then processed within a two-layer GraphSAGE (SAGEConv) framework that was optimized for heterogeneous data. At each layer, the features were combined that came from a node's neighbors as well as the node's features itself, so the model would be able to learn both the local behavior as well as the relational dependencies [6]. Mathematically, this neighborhood aggregation and update mechanism is a two-step process for each node v at each layer k:

First, the feature vectors of the neighboring nodes, $\mathcal{N}(v)$, are aggregated into a single vector, typically via mean-pooling:

$$\mathbf{h}_{\mathcal{N}(v)}^{(k)} \leftarrow \mathrm{MEAN}(\{\mathbf{h}_u^{(k-1)}, \forall u \in \mathcal{N}(v)\}) \tag{1}$$

Second, the node's own embedding from the previous layer, $\mathbf{h}_v^{(k-1)}$, is concatenated with the aggregated neighbor vector. This combined vector is then transformed by a trainable weight matrix $\mathbf{W}^{(k)}$ and a non-linear activation function σ (ReLU) to produce the node's new embedding for the current layer:

$$\mathbf{h}_v^{(k)} \leftarrow \sigma\left(\mathbf{W}^{(k)} \cdot \mathrm{CONCAT}(\mathbf{h}_v^{(k-1)}, \mathbf{h}_{\mathcal{N}(v)}^{(k)})\right) \tag{2}$$

where $\mathbf{h}_v^{(k)}$ represents the feature vector of node v at layer k, and $\mathcal{N}(v)$ denotes the set of its immediate neighbors. This iterative process allows the node embeddings to capture increasingly larger neighborhood structures with each subsequent layer.

The hidden layers were set to 32 dimensions, with the non-linear transformation being carried out by the ReLU activation function. Dropout was then used after each layer to prevent overfitting. The final output layer generates the fraud likelihood scores for the card nodes based on a sigmoid classifier. This framework was decided upon as the best compromise between computational efficiency and high performance on this relational learning task.

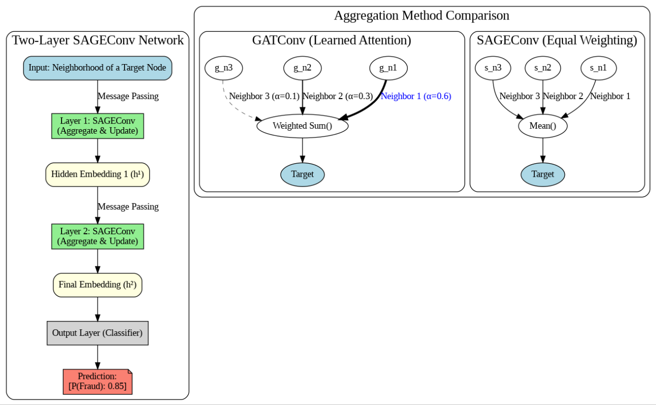
Two-Layer SAGEConv Network
Input: Neighborhood of a Target Node
Message Passing
Layer 1: SAGEConv (Aggregate & Update)
Hidden Embedding 1 (h¹)
Message Passing
Layer 2: SAGEConv (Aggregate & Update)
Final Embedding (h²)
Output Layer (Classifier)
Prediction: [P(Fraud): 0.85]
Aggregation Method Comparison
GATConv (Learned Attention)
g_n3
g_n2
g_n1
Neighbor 3 (α=0.1)
Neighbor 2 (α=0.3)
Neighbor 1 (α=0.6)
Weighted Sum()
Target
SAGEConv (Equal Weighting)
s_n3
s_n2
s_n1
Neighbor 3
Neighbor 2
Neighbor 1
Mean()
Target

Figure 3: GNN Architecture and Message Passing

2.4. *Training Strategy*

Highly skewed nature of fraud detection was very challenging, as there were only a small number of fraudulent transactions in the dataset. To address this skews, the dataset was trained using a weighted cross-entropy loss where the class weights were reversed proportionally to their proportions. In this way, the fraudulent cases received the necessary prominence during optimization. Adam optimizer with weight decay was used to stabilize the convergence and avoid overfitting [7][8]. Node masking was used to divide the dataset into the training, validation, and the test sets so that the evaluation procedure does not leak between the splits. Early stopping based on validation performance was used to avoid unnecessary running time as well as prevent overfitting to the majority class.

3. Experiments and Results

3.1. Experimental Setup

Model performance was measured based on the area under the receiver operating characteristic curve (ROC AUC), as this metric is appropriate for highly imbalanced classification problems since it captures the capability to properly order positive/negative samples. As a baseline, we trained an XGBoost classifier on the same pool of engineered features that were utilized to initialize the nodes, although we presented them as a flattened tabular format instead of a graph. XGBoost is generally accepted as a competitive baseline used in fraud detection as a result of the capability to deal with heterogeneous features as well as nonlinear interactions.

3.2. Performance Evaluation

Comparison between the baseline and the graph-based model showed that the relational method clearly wins. XGBoost baseline gave a ROC AUC of 0.7075 on the test, while the SAGEConv GNN gave a ROC AUC of the order of 0.8943. This almost 0.19 gain shows that graph-based modeling is taking advantage of some very important relational patterns that the transaction-level cannot learn about.

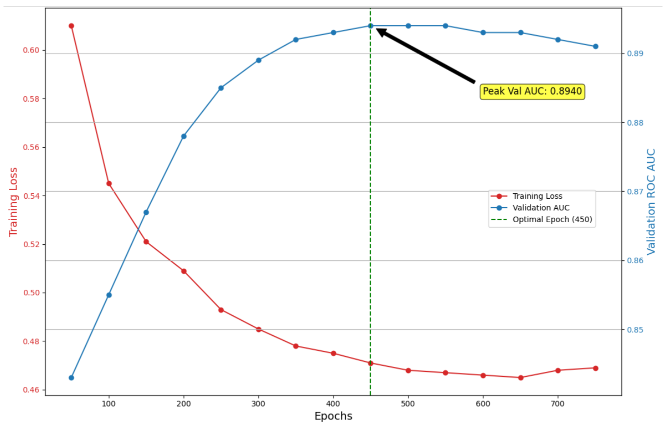

Figure 4: Training Loss and Validation ROC AUC Graph

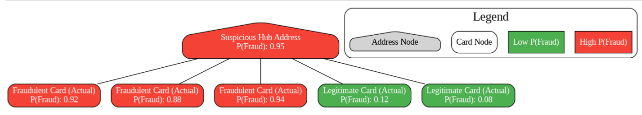

Figure 5: Fraud Pattern Nodal Graph

3.3. Architectural Comparison

To determine whether higher-level architectures would yield additional gains in performance, we also tried Graph Attention Networks (GATConv), that place learnable weights on neighbors at aggregation. While the motivation behind attention mechanisms is attractive, the performance was worse. Training was unstable, the models did not generalize as strongly, reaching ROC AUC less than 0.82 on the test set. These results indicate that increased architectural complexity can bring along tuning difficulties and overfitting possibilities in sparse, highly imbalanced graphs, where the simpler SAGEConv approach balances expressivity more adequately vs. the price offered in additional robustness.

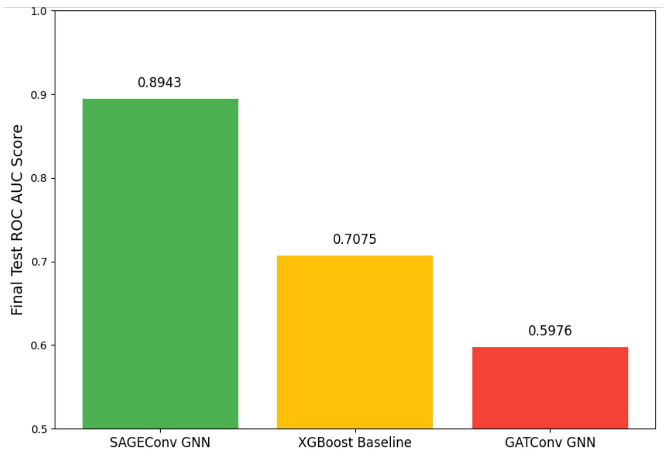

Figure 6: Graph of Multiple Models' ROC AUC Scores

4. Discussion

4.1. Analysis of GNN Performance

The reason why the GNN SAGEConv is effective is that it knows how to summarize contextual information throughout the graph. Taking both the features of a node as well as those of the neighbors into consideration, the structure the model is looking for would otherwise be unseen. A shipping address that is associated with several newly seen cards, or a group of cards that keep bombing the same destination, both point to collusive activities. Models taking transactional snapshots independently cannot see those patterns, reflecting the value add that graph-based relational learning provides to detecting fraud [9].

4.2. Limitations

Despite the positive outcomes, there are also some constraints that must be acknowledged. Our measurements were carried out on a sampling of the dataset, rather than the full body, which would prevent generalizability. Then, the graph schema used cards and addresses, where other players such as merchanting, devices, or IP addresses would engender richer re-relational signals. Finally, the present work stems from offline evaluation; actual working on live-time applications would necessitate the incremental construction of the graph that is efficient too as quick inference processes, problems remaining to be tack-led.

5. Conclusion

This work proposes a heterogeneous GNN-based framework for collusive fraud detection in e-commerce, respecting the explicit relationships between payment cards and ship addresses. By converting transactional data into a graph representation based on relational learning, the model effectively captures the otherwise invisible fraudulent structural patterns that traditional algorithms cannot see [10]. SAGEConv GNN significantly outperformed a competitive XGBoost baseline, demonstrating the significance of connectivity information as a source of discriminative insights. Subsequent work should generalize this framework to more entity classes, scale the tactic to the full dataset, and explore real-time execution strategies. Graph-based relational learning ultimately supplies a powerful paradigm that can be used to fortify fraud prevention within current financial systems.

References

[1] J. Wang, X. Liu, and T. Zhao, “Graph Neural Networks for Financial Fraud Detection: A Review,” *arXiv preprint* arXiv:2411.05815, 2024.

[2] S. Zhang, R. Li, and P. Wang, “Financial fraud detection using graph neural networks: A systematic review,” *Expert Syst. Appl.*, vol. 245, p. 123992, 2024.

[3] L. Chen, Y. Guo, and H. Wang, “GoSage: Heterogeneous Graph Neural Network Using Hierarchical Attention for Collusion Fraud Detection,” in *Proc. 32nd ACM Int. Conf. Inf. Knowl. Manag. (CIKM)*, 2023, pp. 4325–4332.

[4] H. Liu, M. Liang, and C. Sun, “Modeling Heterogeneous Graph Network on Fraud Detection (C-FATH),” in *Proc. 27th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (KDD)*, 2021, pp. 2793–2803.

[5] IEEE-CIS Fraud Detection – Kaggle, “Can you detect fraud from customer transactions?,” Vesta Corporation / IEEE-CIS, Kaggle dataset. [Online]. Available: https://www.kaggle.com/c/ieee-fraud-detection

[6] A. Kumar, L. Fang, and Y. Zhao, “CaT-GNN: Enhancing Credit Card Fraud Detection via Causal Temporal Graph Neural Networks,” *arXiv preprint* arXiv:2402.14708, 2024.

[7] J. Wu, Y. Zhou, and S. Li, “DyHDGE: Dynamic heterogeneous transaction graph for fraud detection,” *Pattern Recognit. Lett.*, vol. 177, pp. 69–78, 2024.

[8] C. Gao, Q. Liu, and Y. Xu, “GE-GNN: Gated Edge-Augmented Graph Neural Network,” *IEEE Trans. Big Data*, vol. 11, no. 4, pp. 981–993, 2025.

[9] F. Li, Z. Huang, and J. Tan, “Graph neural network for fraud detection via context encoding and adaptive aggregation,” *Expert Syst. Appl.*, vol. 239, p. 122950, 2024.

[10] Y. Wang, Y. Liu, and L. Zhang, “Alleviating the Inconsistency Problem of Applying Graph Neural Network to Fraud Detection (GraphConsis),” *arXiv preprint* arXiv:2005.00625, 2020.

[11] D. Chen, S. Zhao, and X. Wang, “Detecting Credit Card Fraud via Heterogeneous Graph Neural Networks with Graph Attention,”