Detecting Financial Fraud Using Deep Learning

Joshua Picchioni
School of Computer Science
University of Windsor
Windsor, Ontario, Canada
picchioj@uwindsor.ca

Arnob Banik
School of Computer Science
University of Windsor
Windsor, Ontario, Canada
banik8@uwindsor.ca

Saumya Buch
School of Computer Science
University of Windsor
Windsor, Ontario, Canada
buchs@uwindsor.ca

Akkhar Ulok
School of Computer Science
University of Windsor
Windsor, Ontario, Canada
ulok@uwindsor.ca

***Abstract*—Detecting financial fraud has become increasingly challenging with improvements in fraud techniques, especially in the age of online transactions. Because of this, traditional machine learning models often struggle to capture the intricate relationships and temporal dynamics within transactional data. To address this, we explore the use of Graph Neural Networks (GNNs) for credit card fraud detection by representing transactions as nodes and behavioural or feature-based similarities as edges in a graph. Using the BankSim dataset, we construct transaction graphs through k-nearest neighbour methods and evaluate four GNN architectures: Graph Convolutional Network (GCN), Graph Attention Network (GAT), GraphSAGE, and Adaptive Sampling and Aggregation-Based GNN (ASA-GNN). Their performance is compared with standard classifiers, including k-Nearest Neighbours (KNN), XGBoost, and Random Forest. This experiment is carried out on both balanced and imbalanced datasets using the synthetic minority oversampling technique (SMOTE) and stratified sampling to balance out the unbalanced nature of transactional datasets. The performance of the models was based on precision, recall, F1 score, and area under the ROC curve (AUC), with a focus on F1 score. The final results show that GAT and GraphSAGE outperform the other models. GAT achieved an F1 score of 92.19% becoming the highest score out of all four GNNs. ASA-GNN, while designed to handle class imbalance, underperformed when applied to the selected dataset. These findings show the future of GNNs in building more robust fraud detection systems by combining the structural and temporal context in transaction data.**

***Index Terms*—Graph Neural Networks, Transaction Fraud Detection, Credit Card Fraud, Graph Representation Learning, Imbalanced Classification.**

I. Introduction

Financial fraud is an ongoing and evolving threat to global economic stability. As digitalisation deepens, fraudulent activities—especially in transactional systems—have become increasingly complex, requiring equally advanced detection methods. In 2022, Canada alone recorded over 91,000 fraud cases, resulting in losses of over $531 million [1].

Traditional machine learning models, such as K-Nearest Neighbors (KNN), XGBoost, and Random Forest, have been widely used to identify fraud and have a high-performance ratio on structured datasets focused on credit card transactions. However, these models fail to capture the complex relational structure embedded within financial systems, for example, among users, merchants, and transactions.

To address such limitations, researchers have shifted increasingly towards Graph Neural Networks (GNNs)—a class of deep learning models that can leverage the topological and behavioral context in graph-structured data. GNNs have tremendous potential in cases where fraud is latent and widespread among connected entities. In such cases, transactions are represented as nodes, whereas connections between entities (e.g., similar transaction patterns, shared accounts) form edges, enabling message-passing mechanisms to detect latent fraud patterns.

Graph neural networks (GNNs) have been used in recent years to learn representations automatically for some prediction tasks [2], [3], [4], [5]. GNN is different from other machine learning methods in that it uses a neighbourhood aggregation method to learn representations and uses a neural network for downstream tasks [6]. It can learn complex interactions between samples and node feature relationships without feature engineering [7]. The research in [8] has used GNN for the transaction fraud detection task. However, it does not use all the features that are available to construct a full graph.

Recent studies demonstrate the higher representational power of GNNs in fraud detection scenarios. For instance, Graph Convolutional Networks (GCNs) [9] employ spectral graph convolutions to learn node neighborhood feature representations, while GraphSAGE [10] offers an inductive method for embedding new nodes using sampling and aggregation. Graph Attention Networks (GATs) [11] extend this idea further by learning attention weights that focus on the most significant neighbors at aggregation.

Building on these improvements, ASA-GNN [12] presents adaptive neighborhood sampling and dynamic aggregation to resolve two common challenges—oversmoothing and neighbor homogeneity—imposed on deeper GNN architectures. In our work, the GNN models have been developed considering the scenario of credit card fraud detection using the BankSim dataset [13], where the effective representation learning ability of these models is evaluated for both balanced and imbalanced conditions, including some cases with over-sampling in synthetic conditions using SMOTE.

The pipeline of experiments encompasses stratified subsampling, graph construction through k-nearest neighbor (k-NN) methods, cross-entropy loss-based training of graph neural network (GNN)-based systems, and extensive testing with measures such as precision, recall, F1 score, and ROC AUC. In addition to this, this paper compares four GNN architectures – graph convolutional network (GCN), GraphSAGE, graph

attention network (GAT), and ASA-GNN, providing inputs into the accuracy, interpretability, and scalability trade-offs for graph-based fraud detection systems. We have also compared these models with the well-known traditional machine learning architecture and specifically under graph sparsity, class imbalance, and hardware limitation conditions.

In this way, we aim to demonstrate that while traditional machine learning models yield high metrics on balanced datasets, GNNs offer a more complex, holistic, and interpretable approach, particularly in uncovering fraud embedded within relational structures.

II. Problem Statement

A. Formal Definition: Fraud Detection via Graph-based GNN

Formal Definition: Fraud Detection via Graph-based GNN

Let

$$T = \{t_1, t_2, \ldots, t_N\}$$

be a set of N transactions, where each transaction t_i is represented by a feature vector in $\mathbb{R}^d$ (e.g., including features such as `step`, `age`, `amount`, etc.). Let

$$Y = \{y_1, y_2, \ldots, y_N\} \subseteq \{0, 1\}$$

be the corresponding set of binary labels, where $y_i = 1$ indicates a fraudulent transaction and $y_i = 0$ a legitimate one.

1. Graph Construction.

Define a function

$$\psi : T \to 2^T$$

that maps each transaction $t_i \in T$ to the subset of transactions

$$\psi(t_i) \subseteq T$$

corresponding to its k-nearest neighbors in the feature space. We then construct an undirected graph

$$G = (V, E)$$

where

$$V = T \quad \text{and} \quad E = \{(t_i, t_j) \mid t_j \in \psi(t_i) \text{ or } t_i \in \psi(t_j)\}.$$

In other words, each transaction node is connected to its k-nearest neighbors based on feature similarity.

2. Fraud Classification Function.

Let

$$f : T \to \{0, 1\}$$

be the function that assigns a true fraud label to each transaction. That is, for each $t_i \in T$, $f(t_i) = 1$ if t_i is fraudulent and $f(t_i) = 0$ otherwise.

3. Graph Neural Network (GNN) Training.

Let Θ denote the parameters of a GNN model operating on G that learns node embeddings $\mathbf{h}_{t_i}$ for each transaction $t_i \in T$. A final classifier C_Θ is then trained to approximate f as follows:

$$\widehat{f}(t_i) = \arg\max_{\ell \in \{0,1\}} C_\Theta\big(\mathbf{h}_{t_i}\big),$$

where $\widehat{f}(t_i)$ is the predicted label for transaction t_i. The GNN is optimized by minimizing the cross-entropy loss over a labeled training subset of T.

B. Motivations

The increasing scale and sophistication of financial fraud represent a significant threat to individuals and institutions. Traditional machine learning methods, while effective on structured data, generally cannot model the complex, relational nature of transactional behaviour underlying fraud networks.

As fraud schemes grow more complicated and networked, the need for models able to untangle these relationships has grown more critical. GNNs offer a new advantageous solution that utilizes the structural and topological regularities in financial transactions. This enables more context-aware and nuanced fraud detection. We are inspired by the possibility of exploring GNN architectures that can learn from transaction graphs—where nodes are single transactions and edges represent behavioural or feature-based similarities. By doing this, we aim to go beyond straightforward classification and toward a more explainable and effective fraud detection system.

Additionally, current research reveals a gap between academic development in GNNs and their practical applicability in fraud detection. We are motivated to bridge this gap by comparing several GNN architectures with traditional models and evaluating their performance on balanced and imbalanced data conditions.

C. Justifications

The selection of applying GNNs here is justified by some compelling reasons:

1) **Relational Modeling Ability:** GNNs inherently incorporate relational relationships among transactions, unlike traditional ML models. Fraud trends, especially in patterns amongst accounts, merchants, or geographic clusters, and GNNs are better positioned to learn and take advantage of this.
2) **Performance Under Class Imbalance:** Datasets for financial fraud are typically imbalanced. GNNs, especially advanced versions like ASA-GNN, possess built-in mechanisms like adaptive neighbour sampling that minimise this issue more effectively than naive oversampling in traditional ML.
3) **Scalability and Generalizability:** Methods like GraphSAGE and GAT are inductive learning algorithms and can generalize easily to unseen nodes or shifting patterns of fraud. This renders them more suitable to real-world application where new information is constantly emerging.
4) **Interpretability and Visual Insights:** By mapping learned embeddings to low-dimensional space (e.g., via t-SNE), GNNs enable better model decision interpretability. Users can visually observe groups of suspicious vs. non-suspicious transactions—a primary need in audit and compliance environments.

5) **Benchmarking Validity:** The comparative design of this study—benchmarking GCN, GraphSAGE, GAT, and ASA-GNN against KNN, XGBoost, and Random Forests—provides a compelling reason for the chosen models and warrants the use of the GNN method across different conditions.

III. Related Works

A. Machine Learning-based Fraud Detection Models

Lee et al. [14] benchmarked five mainstream ML classifiers. It included logistic regression, K-nearest neighbors, SVM, decision tree, and random forest. This was done on an eight-year panel of Indonesian non-financial firms to flag try and determine when financial-statement fraud occurs. Guided by the fraud-triangle theory (which are opportunity, pressure, and rationalization), their feature-selection pipeline highlighted receivable, inventory, and gross-profit ratios as key red flags. From their data, they determined that random forest was the best detector for their dataset. Dependence on tabular accounting ratios however could limit the robustness when the data filings are noisy or incomplete.

Liu et al. [15] built a two-tier detection scheme for Chinese listed companies. This first started with training logistic-regression and gradient-boosting decision-tree screens and then feeding their outputs into a random-forest model to try to determine and flag suspicious statements. On a 100-firm dataset, the stacked model outshone any single learner, offering a lightweight watchdog for market participants and its modest sample and chiefly financial-ratio focus may constrain the transferability.

Yankol et al. [16] frames first-notice-of-loss data as a cost-sensitive learning task, blending numeric, categorical, and NLP-derived text cues to spot suspicious auto-insurance claims at the moment they are opened. By embedding mis-classification costs into gradient-boosted trees and calibrating high cut-off thresholds, the approach steers adjusters toward the claims whose potential payout and fraud risk make investigation financially worthwhile. Its heavy dependence on carrier-specific cost matrices and French regulatory context, however, could curb out-of-the-box transferability.

Afriyie et al. [17] explored a supervised-learning pipeline for credit-card fraud, pitting logistic regression and a single decision tree against a random-forest group on a publicly available (synthetic) U.S. transactions set. After class-rebalancing with undersampling, the forest emerged as the most reliable flagger of suspicious operations, and the authors highlighted risk patterns. The main source of fraud was common with elderly card-holders and late-night (22:00–04:00 GMT); as those purchases were disproportionately hit. Because the underlying data are simulated and U.S.-centric, though, the model's real-world generalizability remains uncertain.

B. GNN-based Fraud Detection Models

Sultana et al. [18] introduced detectGNN, a temporal feature-augmented GNN model specifically designed for credit card fraud detection. It learns temporal relationships in the dependencies of a transaction sequence by adding temporal information into the graph representations. The authors stated that the model significantly improves accuracy in detecting fraud. Its focus on temporals, nevertheless, may effectively limit generalizability in the instance of poor temporal signals.

A more novel approach combining causal inference and temporal GNNs called the CaT-GNN was proposed by Duan et al. [19]. The model leverages causal relationships in sequences of transactions to enhance fraud detection reasoning. Even though leveraging one private and two public datasets showed that CaT-GNN performs better than existing approaches, there was one potential drawback was for the need for domain knowledge for the correct implementation and interpretation of causal inference components.

The incorporation of Reinforcement Learning (RL) into GNNs for dynamic fraud detection was investigated by Dong et al. [20]. To react to new fraud patterns, the model learns to modify its detection tactics in response to environmental feedback. Their method has potential for real-time applications. However, in practice, it is difficult due to the additional complexity that comes with integrating RL.

The semi-supervised graph attention model named SemiGNN came from Wang et al. [21] for identifying financial fraud. The research team developed SemiGNN for detecting financial fraud. financial fraud. Attention mechanisms within the model structure enable it to process data through two core processes. SemiGNN utilizes the graph structure along with user behavior patterns for its operations. The performance evaluation showed that SemiGNN exceeded baseline models regarding accuracy metrics with Alipay user data.

The research by Tian et al. [22] demonstrated ASA-GNN, which uses Adaptive Sampling and an Aggregation-based Graph Neural Network. The system has been created specifically to detect transaction fraud. This model, which includes a contextual dynamic framework that improves its mechanism. The algorithm combines all key features obtained from intelligent neighboring sources. ASA-GNN demonstrated better performance than most models through its results in F1-score and accuracy assessment and used state-of-the-art algorithms on three real-world financial datasets. The use of attention-based neighbour selection techniques delivered better interpretability of results. Better analyst comprehension of detection results became feasible for fraud investigators through the attention-based mechanism. The method depends on selecting high-quality adjacent samples, which occurs within this system because it struggles to handle either sparse or noisy transaction data. A single testing platform of the method makes generalization capabilities uncertain.

IV. Methodology

A. Material and Data

We used the BankSim public dataset "Synthetic data from a financial payment system," [13] an agent-based bank-payment simulator calibrated against aggregated Spanish bank data using statistical and social-network analysis. It was Simulated

over 180 time-steps spanning six months; it contains 594643 transactions of which 587443 are real and 7200 are synthetic generated by "thief" agents perpetrating two frauds a day. Each record includes timestep, merchant and customer IDs, age, gender, ZIP, category, amount, and a binary fraud indicator. The data is kept private while enabling reproducible fraud-detection research.

B. Proposed Methods

We have outlined the pipeline of our GNN-based fraud detection model in the following steps:

1) **Data Preprocessing:**
   a) The raw fraud detection data is loaded. From this data, valuable features (e.g., `step`, `age`, `amount`) are extracted.
   b) The `age` column is cleaned and converted to a numeric format, and rows with missing or invalid values are removed.
   c) Stratified subsampling is performed to select 10,000 records, maintaining the class distribution (fraud vs. legitimate) intact.
   d) Features are normalized (using z-score normalization) to yield a standard measurement scale.
2) **Data Balancing:**
   a) Since there is a class imbalance involved in fraud detection, the SMOTE algorithm is applied to generate new samples for the minority (fraud) class.
3) **Graph Construction:**
   a) The preprocessed features are utilized to build a k-nearest neighbor (KNN) graph. Every transaction in the graph is a node.
   b) Edges are created between a node and its k-nearest neighbors in terms of similarity in the feature space computed by Euclidean distance.
   c) The obtained graph is then converted to a PyTorch Geometric `Data` object-compatible representation, preserving node features and edge connections.
4) **Fraud Detection using GNN:**
   a) A Graph Neural Network (GNN) (GCN, GraphSAGE, GAT, ASA-GNN) is trained on the constructed graph.
   b) The GNN model updates node embeddings recursively through message passing on the graph edges.
   c) There are fraud labels on the transaction nodes alone and no other nodes (if at all present).
   d) A cross-entropy loss function trains the network, and lastly, a binary classifier decides in the final layer whether each transaction is a fraud.
5) **Evaluation:**
   a) The performance of the trained GNN is evaluated using metrics such as Accuracy, Precision, Recall, F1 Score, and ROC AUC.
   b) A threshold sweep is performed on the validation set to determine the optimal decision boundary for fraud classification.

C. Algorithmic Representation of the Fraud Detection Pipeline

The overall fraud detection pipeline consists of two main components, which are described in Algorithms 1–2 below.

Algorithm 1 Construct Graph for Fraud Detection

Input: Raw dataset D with features (`step`, `age`, `amount`),
Fraud labels Y (0 = legit, 1 = fraud),
Number of neighbors k for KNN graph
Output: Graphs $G_{\text{train}} = (X_{\text{train}}, Y_{\text{train}}, E_{\text{train}})$, $G_{\text{test}} = (X_{\text{test}}, Y_{\text{test}}, E_{\text{test}})$

Subsample 10,000 records using stratified sampling on Y
Scale features using z-score normalization
if class imbalance exists **then**
    Apply SMOTE to balance classes
end if
Split the dataset into training and test sets
for each subset in {train, test} **do**
    for each node i in the subset **do**
        Find k nearest neighbors of i using Euclidean distance
        Create directed edges (i, j) to each neighbor j
    end for
end for
Construct graphs:
- $G_{\text{train}} \leftarrow (X_{\text{train}}, Y_{\text{train}}, E_{\text{train}})$
- $G_{\text{test}} \leftarrow (X_{\text{test}}, Y_{\text{test}}, E_{\text{test}})$

if visualization is required **then**
    Convert G_{train} to a NetworkX graph
    Draw using a layout; color nodes by fraud label
end if
return G_{train} and G_{test}

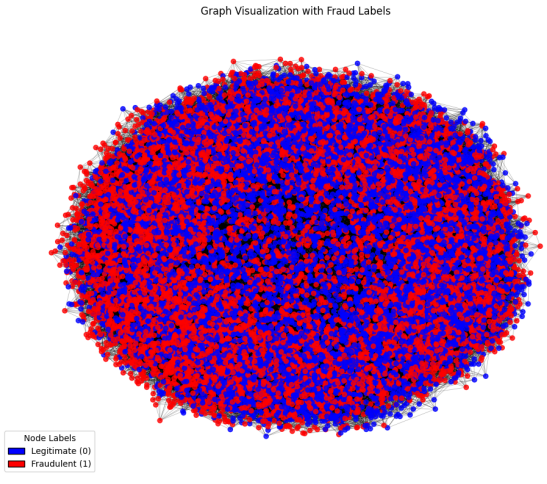

Fig. 1. Dataset Graph Visualization

Algorithm 1: Construct Graph for Fraud Detection.

We load the BankSim dataset [13], convert `age` to numeric and remove invalid entries, then stratify and sample 10,000 records by fraud rate. After z-score scaling and SMOTE balancing, the data are split, a k-nearest-neighbor graph $G = (V, E)$ is constructed by linking each node to its k most similar peers (Euclidean distance). Then the final graphs G_{train} and G_{test} are stored in PyTorch Geometric format. Figure 1 illustrates a visualization of the graph dataset, with labels indicating legitimate and fraudulent transactions.

Algorithm 2 Train and Evaluate GNN for Fraud Detection

Input: $G_{\text{train}} = (X_{\text{train}}, Y_{\text{train}}, E_{\text{train}})$, $G_{\text{test}} = (X_{\text{test}}, Y_{\text{test}}, E_{\text{test}})$, chosen GNN model (e.g., GCN, GraphSAGE, GAT), hyperparameters: learning rate η, hidden dimension, number of epochs T.

Output: trained model M^*, best threshold θ^*, final evaluation metrics.

Create GNN model M with input dimension = $\dim(X_{\text{train}})$
Initialize optimizer (Adam) and loss (Cross-Entropy)
for each epoch $t = 1$ to T **do**
    Set M to training mode
    $\hat{Y} \leftarrow M(X_{\text{train}}, E_{\text{train}})$ ▷ Forward pass
    $\mathcal{L} \leftarrow \text{CrossEntropy}(\hat{Y}, Y_{\text{train}})$ ▷ Compute loss
    Backpropagate and update model parameters
    (Optional) Log training metrics
end for
Set M to evaluation mode
Compute predicted probabilities $\hat{P}$ on G_{test}
for each threshold $\theta \in [0, 1]$ **do**
    Convert $\hat{P}$ to binary predictions $\hat{Y}_\theta$
    Compute F1 score for $\hat{Y}_\theta$
end for
$\theta^* \leftarrow \arg\max_\theta \text{F1}(\hat{Y}_\theta)$
Compute final metrics (Accuracy, Precision, Recall, F1, ROC AUC) using θ^*
return trained model M^* and best threshold θ^* with evaluation metrics

Algorithm 2: Training and Evaluation of GNN.

We construct a GNN M (e.g. GCN, GraphSAGE, GAT) whose input layer is the same size as node features and whose output is a linear classifier. Training runs for T epochs on G_{train} with Adam and cross-entropy: within each epoch, messages propagate along edges to update embeddings, the loss is computed, and gradients update the model. After training, we switch M to evaluation mode, apply it in G_{test} to obtain softmax scores, and sweep a threshold $\theta \in [0, 1]$ in order to perform binary predictions. The optimal threshold θ^* that maximizes the F1 score is selected, and afterwards we report accuracy, precision, recall, F1, and ROC AUC at θ^*.

To evaluate the efficacy of graph-based fraud detection, we implement and compare four Graph Neural Network (GNN) variants:

1) Graph Convolutional Network (GCN): The GCN [9] performs spectral graph convolutions by aggregating and linearly transforming the normalized sum of a node's features and those of its immediate neighbors. Its simplicity and efficiency make it a strong baseline for semi-supervised node classification.

2) GraphSAGE: GraphSAGE [10] extends GCN by using learned, sample-and-aggregate functions (e.g., mean, pooling, or LSTM aggregators) to inductively generate node embeddings. At each layer, GraphSAGE samples a fixed number of neighbors, enabling scalability to large, dynamic graphs.

3) Graph Attention Network (GAT): GAT [11] introduces self-attention to GNNs, computing learnable attention coefficients for each node–neighbor pair. By weighting neighbor contributions adaptively, GAT can distinguish the relative importance of different neighbors during message passing.

4) Adaptive Sampling and Aggregation GNN (ASA-GNN): ASA-GNN [12] enhances conventional GNNs with two novel mechanisms: (1) *Adaptive Neighbour Sampling*, which selects high-quality and oversamples fraud-related neighbors using cosine similarity and edge weights (Eqs. 7–8 in [12]); and (2) an *Adaptive Aggregation Layer*, which employs a neighbor-diversity entropy gate (Eq. 16 in [12]) and time-decayed attention (Eq. 11 in [12]) to mitigate oversmoothing and camouflage effects.

Altogether, these procedures represent a single pipeline that combines graph-based representation learning with adaptive sampling and aggregation for efficient fraud detection.

D. Conditions and Assumptions

When creating and validating our graph-based fraud detection pipeline, we assume these below-stated key assumptions:

1) **Constant Graph Structure:** Once we create the k-nearest-neighbor graph, its structure is maintained unchanged at training as well as testing times. No additional nodes or edges are introduced later on.
2) **Clean Feature Set:** We assume all the remaining transaction records after preprocessing give a complete, valid feature vector; records with missing fields required have already been prefiltered.
3) **Reliable Labels:** All the binary fraud indicators are derived from the simulator without deliberate corruption or noise, so we take them as ground truth.
4) **SMOTE Fidelity:** Fraud samples generated synthetically using SMOTE are executed to truly represent minority-class real-world patterns without adding deceptive artifacts.
5) **Local Similarity:** Nearby transactions within the chosen feature space (using Euclidean distance or cosine similarity) are presupposed to likely share the same fraud status and hence are deserving to be united and merged collectively.
6) **Fixed Hyperparameters:** We use constant values for graph degree k, per-layer sample budgets $\hat{z}_k$, learning rate, hidden dimensions, and network depth, assuming that such configurations remain valid for various data splits and simulation runs.

E. Formal Complexity

Let $N = |V|$ be the siZe of transactions, d the feature dimension, D the average degree of nodes, K the number of GNN layers, F the size of hidden-state, $\hat{z}$ the budget of neighbour-sampling, and $|E|$ the number of edges. Then,

1) **Preprocessing & SMOTE:** We derived $O(Nd)$ for synthetic sample creation and z-score scaling, and $O(N \log N)$ for stratified subsampling.
2) **Construction of k-NN Graph:** $O(N^2 d)$ for exact nearest neighbours, or $O(N \log N\, d)$ with approximate solutions.
3) **GNN Training (epoch by epoch):** $O\big(K\,(|E|\,F + N\,F^2)\big)$ for message passing and linear transformations.
4) **ASA-GNN Neighbour Sampling:** $O\big(K\,N\,(D\,d\; + D \log \hat{z})\big)$ to compute cosine-similarities (cost $O(d)$ per neighbour) and select the top $\hat{z}$ entries.
5) **ASA-GNN Aggregation & Gating:** $O\big(K\,(|E|\,F + N\,D)\big)$ to compute attention calculation (cost $O(F)$ per edge) and entropy-based gating (time complexity $O(D)$ per node).

Typically, the costliest term in typical situations is the training term of the GNN $O(K\,(|E|\,F + N\,F^2))$, especially when $|E|$ and F are large.

These analyses ensure that our method is scalable to typical set sizes of data and yet achieves high fraud-detection accuracy.

V. Computational Experiments

A. What Experiments?

The main experiments conducted used the simulated datasets mentioned in Methodology that were transformed into KNN based graphs, and the ASA-GNN was based on it's own modification to the homogeneous graphs. The performance of those four GNNs on the dataset was then compared to the performance of previous works that used ML models on the same dataset giving us an accurate determination on the effectiveness of GNN-based models in transactional fraud classification.

Part of the experimentation was to transform the original dataset, which was converted to a GNN graph, into heterogeneous input data. This heterogeneous input was then also transformed into a homogeneous graph where each node represented a transaction and edges represented connections between the transactions. The transformation pipeline involved the following stages:

1) **Transaction Graph Conversion:** We first converted the heterogeneous graph structure, comprising entities such as users, merchants, and terminals, into a pure transaction-level graph. Each node in this graph corresponds to a unique transaction instance. This flattens the heterogeneous relationships into transaction-centric records.
2) **Adjacency Construction:** An initial undirected edge list was constructed where edges represent heuristic or behaviour-based connections between transactions (e.g., shared user ID, merchant proximity, or temporal linkage). Alongside the edge list, edge attributes were derived to encode connection strengths or similarities—the reduced set of transactions served as the final transaction nodes retained after filtering or de-duplication.
3) **Feature Matrix and Label Preparation:** The next step was to extract node features and binary fraud labels for all nodes. The features typically include transaction amount, time step, merchant code, and other relevant dimensions, depending on preprocessing. The labels form the supervised target for binary classification.
4) **ASA-GNN Edge Refinement:** The core of the ASA-GNN graph transformation involved refining the initial edge structure using adaptive neighbour selection. The edges were redefined by selecting the top-k most behaviorally similar neighbours (k=20) for each node. Cosine similarity was computed between node feature vectors, and edge weights were exponentiated to emphasize highly similar pairs while suppressing noise. The output included:
   a) edge_index_asagnn: filtered and directional edge list specific to ASA-GNN
   b) edge_weight_asagnn: computed similarity-based edge weights
5) **Final Graph Object:** The final homogeneous graph was structured as a torch_geometric.data.Data object, containing:
   a) x: the node feature matrix
   b) y: binary fraud labels for classification
   c) edge_index: the sparse edge list (ASA-GNN refined)
   d) edge_attr: edge weights representing behavioural similarity

The ASA-GNN dataset did not utilize any class imbalance methods, as the model is designed to handle class imbalances and incorporates oversampling into its methods, as shown above.

Following this dataset preprocessing, each model was then trained on the homogeneous graphs created.

The KNN-based graphs consisted of 15808 training samples and 3952 test samples. During the training of GraphSAGE, GAT and GCN, grid search was employed to discover the best hyperparameters for the model, and the best threshold for classifying between fraudulent and non-fraudulent transactions as shown by Algorithm 2 and described earlier.

The hyperparameters being trained and evaluated consisted of the following values: Learning Rates: {0.001,0.005,0.01} Hidden Layer Dimensions: {16,32,64}

The highest value for that configuration of hyperparameters and threshold was then stored if the F1 score was better than that of previous hyperparameter configurations and probability thresholds. Table 1. shows an output of the best hyperparameters and thresholds for each model.

TABLE I
HYPERPARAMETER TUNING OUTPUTS FOR GNN MODELS

Model	Learning Rate	Hidden Dimensions	Threshold
GraphSAGE	0.01	32	0.37
GAT	0.01	16	0.31
GCN	0.01	32	0.37

The model was then reevaluated using the optimal configuration and threshold with the testing data to produce a final output of the results.

The ASA-GNN was trained on a training, evaluation and testing set that was a little different than the other models. All the training methods, hyperparamaters and more were following the methods described in the article for ASA-GNN[12]. The homogeneous graph created for ASA-GNN included 36,572 different nodes. Of these, 21,943 were training nodes, 7,314 were validation nodes, and 7,315 were testing nodes. The separation of nodes for training, testing and validation were highlighted using train, test and validation masks.

The model was trained for 30 epochs, and after every five epochs, it was evaluated on the validation set to ensure that overfitting was not occurring. After 30 epochs, the model was evaluated on the testing set to produce a final output of results and metrics for comparison.

B. What Evaluation Metrics?

Model performance evaluation in the field of credit card fraud detection necessitates moving beyond conventional accuracy-based criteria. Since there are far more genuine transactions than fraudulent ones, identifying them is by its very nature a high-class-imbalance problem. Due to this, a model can attain high accuracy by merely identifying every transaction as authentic, but it will fall short of its intended purpose.

In order to tackle this, we implemented an extensive collection of assessment metrics and visualization techniques, each chosen for its unique pertinence to the difficulties of fraud analysis:

1) Classification Metrics: The **accuracy**, while being the most general measure of correct predictions, is insufficient for imbalanced datasets, such as credit card fraud. A model that always predicts "legitimate" may still score over 98% accuracy but completely fail to detect fraud. Hence, accuracy is reported but not prioritised in model selection.

In fraud detection, false positives can lead to blocked cards, declined purchases, and a negative customer experience. High **precision** ensures that only genuinely suspicious transactions are flagged, reducing unnecessary disruption for cardholders.

Recall (Sensitivity) is critical because missing a fraudulent transaction (false negative) directly translates to financial loss and risk exposure. High recall is necessary to minimize undetected fraudulent activity.

The **F1 Score** is the harmonic mean of precision and recall. It serves as a balanced metric that reflects both the cost of false positives and the cost of false negatives. In the context of fraud detection, where both types of errors have consequences, the F1 score is a primary evaluation criterion during model and threshold selection.

2) Classification Report and Confusion Matrix: To interpret the performance breakdown more granularly, we include:

1) A classification report, showing class-wise precision, recall, F1 score, and support. This helps determine if the model is truly learning to detect the minority fraud class or is biased toward the majority class.
2) A confusion matrix, visualizing the counts of:
   a) True Positives (TP): Fraud correctly identified
   b) False Positives (FP): Legitimate transactions incorrectly flagged
   c) True Negatives (TN): Legitimate transactions correctly identified
   d) False Negatives (FN): Fraud missed by the model
3) In fraud analysis, the confusion matrix is invaluable for identifying the trade-off between financial risk (FN) and customer inconvenience (FP). For instance, a high number of false positives might suggest an overly aggressive model, while a high false negative count would indicate operational risk exposure.

3) t-SNE Embedding Visualization: Finally, we employed t-distributed Stochastic Neighbour Embedding (t-SNE) to visualize the node embeddings generated by each GNN model's final hidden layer. Only the ASA-GNN did not have a t-SNE embedding, as the graph used was slightly different than the KNN-based graph. This technique projects high-dimensional learned features into a 2D space, enabling:

1) Visualization of how well-separated fraudulent vs. legitimate nodes are in the learned representation space
2) Identification of overlap or confusion regions (e.g., where false positives and false negatives occur)
3) Enhanced model interpretability for stakeholders, especially in explaining why certain transactions were flagged

This form of unsupervised visual diagnostics is particularly important in fraud detection applications, where trust, transparency, and accountability are as critical as performance.

C. Implementation Details

All experiments were conducted on Google Colab, using a single-session cloud environment (approx. 12GB RAM, Tesla T4 GPU). Due to hardware limitations, we carefully optimized our pipeline to reduce memory consumption and training time. The ASA-GNN was experimented with using the same method, except using 51 GB of RAM instead of 12 GB.

Software Stack:

1) Python 3.10
2) PyTorch (deep learning framework)
3) PyTorch Geometric (PyG) for GNNs
4) scikit-learn for preprocessing and evaluation
5) Matplotlib and Seaborn for visualization
6) SMOTE for oversampling

D. Results

To see how well different graph neural network models could detect credit card fraud, we tested four architectures:

GraphSAGE, GAT, GCN, and ASA-GNN. Each model was trained on a graph built from a SMOTE-balanced version of the transaction data, then evaluated on a separate test set. We measured their performance using four main metrics—accuracy, precision, recall, and F1 score—which are summarized in Table 2.

TABLE II
GNN MODEL METRICS

Model	Accuracy	Precision	Recall	F1 Score
GraphSAGE	0.9378	0.9152	0.9464	0.9372
GAT	0.9248	0.9012	0.9511	0.9255
GCN	0.9195	0.8915	0.9443	0.9272
ASA-GNN	0.9193	0.7751	0.8251	0.7993

1) GraphSAGE: GraphSAGE performed the best overall out of the four GNN models that were assessed, with an accuracy of 93.78% and F1 score of 93.72%. Even with a few noisy connections between transaction nodes, the model was able to function effectively thanks to its inductive sampling and aggregation approach. It also demonstrated strong recall of 94.64% and precision of 91.52%, showing the ability to detect fraudulent transactions while minimizing false positives.

2) GAT: In terms of performance, GAT trailed GraphSAGE. With a high recall of 95.11%, this model outperformed all other models evaluated. This results in GAT being very effective at identifying fraudulent situations. However, it also seemed more prone than GraphSAGE to mistakenly label valid transactions as fraudulent, as shown by its lower precision of 90.12%.

3) GCN: The GCN model showed competitive results with 92.72% F1 score. This model remains a reliable and straightforward technique for detecting fraudulent transactions, although performing somewhat lower than GAT and GraphSAGE.

4) ASA-GNN: In contrast to all three previous models, ASA-GNN underperformed, achieving an F1 score of 79.93%, a precision of 77.51%, and a recall of 82.51%. While the model was specifically designed to handle class imbalance and camouflage behaviour through adaptive sampling and oversampling, its advantages did not materialize in this experiment. This can be attributed to the use of a homogeneous transaction graph rather than the heterogeneous structure for which ASA-GNN was designed.

Additionally, the need to cap the number of edges due to hardware constraints may have limited the model's access to critical neighbourhood information. As a result, ASA-GNN generated more false positives and missed more fraud cases than the baseline GNNs.

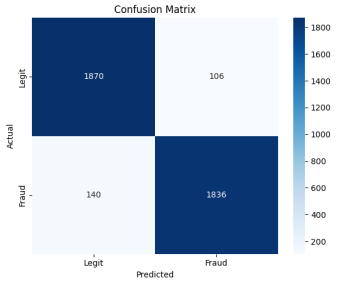

Fig. 2. GraphSAGE Confusion Matrix

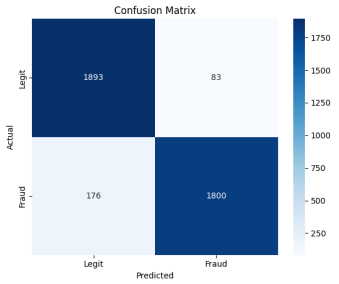

Fig. 3. GAT Confusion Matrix

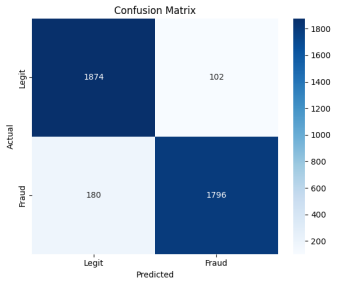

Fig. 4. GCN Confusion Matrix

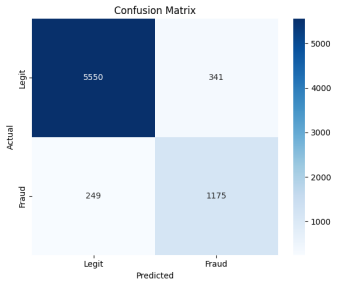

Fig. 5. ASA-GNN Confusion Matrix

This is further evidenced by the confusion matrices for each GNN model. Figures 2 and 3 show that GraphSAGE and GAT achieved the best trade-offs, maintaining high true positives while limiting false positives. Figure 4 shows that GCN was also effective, although it exhibited a slightly higher false positive rate than GAT. However, ASA-GNN generated the

most false positives, which lowered its precision, as shown in Figure 5.

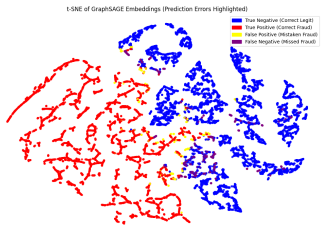

Fig. 6. GraphSAGE t-SNE Plot

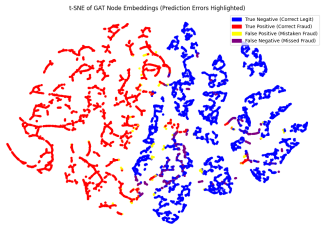

Fig. 7. GAT t-SNE Plot

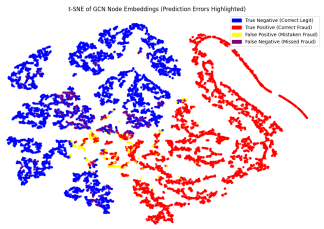

Fig. 8. GCN t-SNE Plot

Figures 6, 7, and 8 present the t-SNE visualizations of the transaction embeddings learned by GCN, GAT, and GraphSAGE, respectively. In Figure 6, the GCN model separates transactions into two main clusters: legitimate (blue) and fraudulent (red). However, many of the errors, such as false positives (yellow) and false negatives (purple), appear around the edges of these clusters, indicating that while the model captured the general boundary between classes, it struggled with borderline cases.

In Figure 7, the GAT visualization appears more dispersed. Although the model successfully identifies a number of fraudulent transactions, the scattered presence of false negatives within the legitimate cluster suggests it had more difficulty distinguishing certain types of fraud. In contrast, Figure 8 demonstrates that GraphSAGE learned a much clearer separation between the two classes. Legitimate and fraudulent transactions form well-defined, compact clusters, with only a few errors located away from these main groups.

Overall, these visualizations reinforce the performance results: GraphSAGE not only achieved the best metrics but also developed the most reliable representation of transaction behavior.

E. Discussions

1) Performance of Graph Neural Networks: Overall, the results affirm that while specialized architectures like ASA-GNN hold potential, simpler, well-tuned models such as GraphSAGE and GAT can outperform them in controlled environments with homogeneous data and constrained computational resources. Again, **ASA-GNN**—although specifically built for fraud detection in financial transaction networks—underperformed with an F1 score of 79.93%. This can be attributed to:

1) Its dependence on a more complex graph structure (heterogeneous nodes/edges) is not fully leveraged in this experiment
2) Edge truncation due to hardware constraints
3) Increased false positive rates are impacting precision

While ASA-GNN remains theoretically promising, its current results highlight that model effectiveness depends on how well the implementation context matches architectural design assumptions.

Now, to further evaluate the efficacy of GNNs against simpler ML models, F1 Score and accuracy are used as metrics to compare with the ML models

2) Comparison with Traditional Machine Learning Models: The data for the performance of the ML models was obtained from previous work using the same dataset. For easier comparison, Accuracy and F1 score were isolated for in Table 3.

TABLE III
ML MODEL METRIC SCORES

ML Model	Accuracy (%)	F1 Score (%)
KNN	99.37	99.00
XGBoost	99.58	99.00
Random Forest	96.21	97.00

These models achieved exceptionally high accuracy and F1 scores, especially XGBoost and KNN, with near-perfect values above 99%. However, these results should be interpreted with caution:

1) ML models operate on tabular data and benefit heavily from manual feature engineering; however, they lack the relational context that GNNs natively incorporate (e.g., transaction similarity, shared behaviours).
2) GNNs, although slightly behind in raw performance metrics, offer better generalizability in dynamic fraud environments where relational behaviour evolves.

3) Interpretation and Trade-offs:

1) GNNs offer structural insights: Their core advantage lies in learning fraud patterns based on network structure and behavioural context, especially useful in fraud scenarios with identity masking and transaction camouflage.
2) ML models offer high predictive power: Their simplicity, speed, and impressive baseline performance make them attractive for quick deployment.
3) ASA-GNN needs better graph structure: While underperforming here, ASA-GNN is expected to excel in more

complex graphs where its oversampling and adaptive attention can better utilize multitype node relationships.

The dataset also plays an important part in performance. First, we attempted to train our models on the IBM Credit Card Transaction Dataset, which contains over 24 million transactions, but only approximately 30,000 fraudulent instances (approximately 0.1%). Utilizing this datset would have posed several computational challenges.

Despite testing various techniques (including downsampling, SMOTE, and architectural modifications), the models were unable to attain a higher precision than 0.14%, even when mirroring prior work. Because of this, we transitioned to a more tabular dataset (BankSim), which allowed for more realistic evaluation and clearer model differentiation.

However, this simplified dataset may have disproportionately favoured ML models, which are highly efficient for tabular features. In contrast, GNNs benefit more when relational information is in complex graphs. As such, the performance of ML models in this experiment reflected the characteristics of the training and testing data more efficiently.

4) Final Remarks: Despite traditional ML models showing higher raw metrics in this case, GraphSAGE and GAT rivalled them closely, offering interpretable, scalable, and behaviour-aware fraud detection capabilities. Future work should aim to:

1) Benchmark all models on identical pipelines with consistent preprocessing.
2) Extend GNN experimentation to heterogeneous transaction graphs.
3) Explore model robustness under evolving fraud strategies.

This evaluation demonstrates that graph-based learning is not only competitive but also essential in fraud detection tasks where relationships are as important as features.

VI. Conclusion

A. Summary

The project confirmed that graph-based deep-learning models can be trained successfully for credit-card fraud detection, but the evidence remains inconclusive as to whether they can outperform machine-learning models. As we used the BankSim dataset (a comparatively small and orderly dataset) any performance advantage may merely reflect the dataset's simplicity rather than a genuine methodological edge. However, the work has deepened our understanding of Graph Neural Networks, improved our skills in PyTorch Geometric, and demonstrated how making end-to-end fraud-detection pipelines on limited cloud resources could still be possible.

B. Future Research

More tests and trials in testing other GNN variants such as GIN and related models, tinkering more with parameters (such as how many Epochs, how much the training–test split is done, etc.). Also, we plan to test robust data-balancing strategies (such as overtraining) to mitigate some of the extreme class imbalances that many banking datasets (including the one we chose) have. We also plan to use faster hardware and software-level optimisations, as they are needed to cut training time and make large-scale experiments practical.

C. Open Problems

We had challenges that remain open problems during our study. First, very large, highly imbalanced datasets (such as the 24-million-transaction IBM dataset) where too demanding for the current workflow, both computationally and in terms of model precision. Then the repeated time-outs and crashes on Google Colab made it clear that relying on free cloud notebooks for heavy graph processing is not a good idea. Fixing this issue is relatively simple (buying more expensive hardware or buying a more powerful cloud service), however requires spending money. Then, finally, converting the raw transactions into a graph representation was very resource-intensive, requiring repeated environment rebuilds and dependency installations after each Colab session reset. Fixing these open problems (scalability, environmental stability, and preprocessing overhead) in a real-world setting where fraud detection goes beyond our study is extremely important.

References

[1] D. Cheng, Y. Zou, S. Xiang, and C. Jiang, "Graph neural networks for financial fraud detection: a review," *Frontiers of Computer Science*, vol. 19, no. 9, pp. 1–15, 2025.

[2] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 922–929.

[3] Q. Li, T. Zhang, C. P. Chen, K. Yi, and L. Chen, "Residual gcb-net: Residual graph convolutional broad network on emotion recognition," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 15, no. 4, pp. 1673–1685, 2022.

[4] X.-h. Wang, T. Zhang, X.-m. Xu, L. Chen, X.-f. Xing, and C. P. Chen, "Eeg emotion recognition using dynamical graph convolutional neural networks and broad learning system," in *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2018, pp. 1240–1244.

[5] T. Song, W. Zheng, P. Song, and Z. Cui, "Eeg emotion recognition using dynamical graph convolutional neural networks," *IEEE Transactions on Affective Computing*, vol. 11, no. 3, pp. 532–541, 2018.

[6] H. Yifan, Z. Jian, C. James, M. Kaili, M. R. TB, C. Hongzhi, and Y. Ming-Chang, "Measuring and improving the use of graph information in graph neural network," in *The Eighth International Conference on Learning Representations (ICLR 2020), Addis Ababa*, 2020.

[7] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters," in *Proceedings of the 29th ACM international conference on information & knowledge management*, 2020, pp. 315–324.

[8] A. Khazane, J. Rider, M. Serpe, A. Gogoglou, K. Hines, C. B. Bruss, and R. Serpe, "Deeptrax: Embedding graphs of financial transactions," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE, 2019, pp. 126–133.

[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017. [Online]. Available: https://arxiv.org/abs/1609.02907

[10] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30. Curran Associates, Inc., 2017, pp. 1024–1034.

[11] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lió, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations (ICLR)*, 2018. [Online]. Available: https://arxiv.org/abs/1710.10903

[12] Y. Tian, G. Liu, J. Wang, and M. Zhou, "Asa-gnn: Adaptive sampling and aggregation-based graph neural network for transaction fraud detection," *IEEE Transactions on Computational Social Systems*, vol. 11, no. 3, pp. 3536–3548, 2024.

[13] Edgar Lopez-Rojas, "Synthetic data from a financial payment system," Kaggle Dataset: https://www.kaggle.com/ealaxi/banksim1, 2025, accessed: 2025-04-16.

[14] C.-W. Lee, M.-W. Fu, C.-C. Wang, and M. I. Azis, "Evaluating machine learning algorithms for financial fraud detection: Insights from indonesia," *Mathematics*, vol. 13, no. 4, p. 600, 2025.

[15] X. Liu, "Empirical analysis of financial statement fraud of listed companies based on logistic regression and random forest algorithm," *Journal of Mathematics*, vol. 2021, no. 1, p. 9241338, 2021.

[16] M. Y. Schalck, "Auto insurance fraud detection: Leveraging cost sensitive and insensitive algorithms for comprehensive analysis," *Insurance: Mathematics and Economics*, vol. 122, pp. 44–60, 2025.

[17] J. K. Afriyie, K. Tawiah, W. A. Pels, S. Addai-Henne, H. A. Dwamena, E. O. Owiredu, S. A. Ayeh, and J. Eshun, "A supervised machine learning algorithm for detecting and predicting fraud in credit card transactions," *Decision Analytics Journal*, vol. 6, p. 100163, 2023.

[18] I. Sultana, S. M. Maheen, N. Kshetri, and M. N. F. Zim, "detectgnn: Harnessing graph neural networks for enhanced fraud detection in credit card transactions," *arXiv preprint arXiv:2503.22681*, 2025.

[19] Y. Duan, G. Zhang, S. Wang, X. Peng, W. Ziqi, J. Mao, H. Wu, X. Jiang, and K. Wang, "Cat-gnn: Enhancing credit card fraud detection via causal temporal graph neural networks," *arXiv preprint arXiv:2402.14708*, 2024.

[20] Y. Dong, J. Yao, J. Wang, Y. Liang, S. Liao, and M. Xiao, "Dynamic fraud detection: Integrating reinforcement learning into graph neural networks," in *2024 6th International Conference on Data-driven Optimization of Complex Systems (DOCS)*. IEEE, 2024, pp. 818–823.

[21] D. Wang, J. Lin, P. Cui, Q. Jia, Z. Wang, Y. Fang, Q. Yu, J. Zhou, S. Yang, and Y. Qi, "A semi-supervised graph attentive network for financial fraud detection," in *2019 IEEE international conference on data mining (ICDM)*. IEEE, 2019, pp. 598–607.

[22] Y. Tian, G. Liu, J. Wang, and M. Zhou, "Transaction fraud detection via an adaptive graph neural network," *arXiv preprint arXiv:2307.05633*, 2023.