Jurnal ELTIKOM :
Jurnal Teknik Elektro, Teknologi Informasi dan Komputer

Vol. 10, No. 1, June 2026, page. 88-98
ISSN 2598-3245 (Print), ISSN 2598-3288 (Online)
DOI: http://doi.org/10.31961/eltikom.v10i1.1497
Available online at http://eltikom.poliban.ac.id

COMPARATIVE ANALYSIS OF GRAPH NEURAL NETWORKS FOR FRAUD DETECTION

Ricky Maulana Fajri[1*], Muhammad Gald Teary[1], Ni Wayan Priscila Yuni Praditya[2]
[1)] Department of Computer System, Indo Global Mandiri, Palembang Indonesia
[2)] Department of System Information, Indo Global Mandiri, Palembang Indonesia
e-mail: rickymaulanafajri@uigm.ac.id, m.gald@uigm.ac.id, niwayanpris@uigm.ac.id@email.com

Received: 4 April 2025 – Revised: 30 March 2026 – Accepted: 13 April 2026

ABSTRACT

Detecting financial fraud is a complex and evolving challenge, particularly because of the relational nature of transaction data, graph sparsity, and severe class imbalance. To the best of our knowledge, this study represents one of the first systematic benchmarks of five prominent Graph Neural Network (GNN) architectures, GCN, GAT, GraphSAGE, GIN, and SGCN, for fraud detection under balanced and imbalanced conditions across multiple public datasets. We explicitly evaluate the impact of the Synthetic Minority Oversampling Technique (SMOTE) on graph-based fraud detection performance, an aspect that has rarely been addressed in prior research. The comparative analysis considers predictive performance (AUC, F1-Score, Precision, Re-call) and computational efficiency to provide actionable guidance for real-world development. The experimental results show that GraphSAGE offers the best trade-off between accuracy and execution time for laten-cy-sensitive environments, while GAT's attention mechanism supports offline, interpretability-driven analysis. These findings provide empirical evidence to inform GNN selection strategies for scalable and effective fraud detection systems.

Keywords*: fraud analysis, graph neural networks, imbalanced dataset, SMOTE.*

I. INTRODUCTION

FRAUD detection in financial transactions is a critical challenges for banks, financial institutions, and online marketplaces [1], [2], [3]. Structured data plays a pivotal role in fraud detection, as it captures important relationships between entities such as users, accounts, and transactions. These relational patterns often help identify fraudulent activities, particularly in large-scale financial ecosystems where anomalies may be subtle and context-dependent. Although traditional rule-based and statistical techniques have been widely adopted, their effectiveness decreases as fraud patterns become more adaptive and complex. Machine learning methods have emerged as a robust alternative because they can learn discriminative patterns from transactional data [4], [5], [6], [7]. Among these approaches, Graph Neural Networks (GNNs) have gained attention due to their ability to model complex relationships between entities, such as users, transactions, and accounts. By using graph structures, GNNs can capture hidden patterns that conventional methods may overlook, making them a promising solution for financial fraud detection.

Despite the success of GNNs in fraud detection [8], [9], [10], [11], [12], [13], [14], existing research has primarily focused on individual models without extensive comparative studies. Most studies show the effectiveness of GNNs in detecting fraud by leveraging relational data, but they often analyze only one model or a limited set of architectures. Moreover, key challenges such as class imbalance, adversarial robustness, and computational efficiency remain underexplored. Although some works propose model enhancements, systematic benchmarking across multiple GNN architectures using real-world fraud datasets remains limited. This gap highlights the need for a deeper investigation into the comparative performance of different GNN models for fraud detection.

To address these research gaps, this study provides a comprehensive comparison of various GNN architectures applied to financial fraud detection. Unlike prior research that focuses on isolated models,

$$h_v^{(l+1)} = \sigma\left(W^{(l)} \sum_{u \in \mathcal{N}(v)} f(h_u^{(l)}, h_v^{(l)}, e_{uv})\right) \quad (1)$$

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2}\, \tilde{A}\tilde{D}^{-1/2}\, H^{(l)}\, W^{(l)}\right) \quad (2)$$

$$H = \tilde{D}^{-1/2}\, \tilde{A}\tilde{D}^{-1/2}\, XW \quad (3)$$

$$\alpha_{uv} = \frac{\exp\left(leakyReLU(a^T[Wh_u \| Wh_v]\right)}{\sum_{\in \mathcal{N}(v)} \exp\left(leakyReLU(a^T[Wh_u \| Wh_k]\right)} \quad (4)$$

$$H^{(l+1)} = \sigma\left(W^{(l)} \cdot AGG\left(\{h_u^{(l)}, \forall u \in \mathcal{N}(v)\}\right)\right) \quad (5)$$

$$h_v^{(l+1)} = MLP^{(l)}\left((1+\epsilon)h_v^{(l)} + \sum_{u \in \mathcal{N}(v)} h_u^{(l)}\right) \quad (6)$$

this work systematically evaluates multiple GNN variants, including Graph Convolutional Networks (GCNs) [15], Graph Attention Networks (GATs) [16], GraphSAGE [17], Graph Isomorphism Networks (GINs) [18], and Simplified Graph Convolutional Networks (SGCNs) [19]. Furthermore, key performance metrics such as accuracy, recall, and computational cost are examined to assess their practical feasibility in real-world fraud detection scenarios. Thus, the contributions of this study are threefold.

1) A systematic comparison of five GNN models across multiple fraud datasets.
2) An explicit evaluation of SMOTE's effect on graph-based fraud detection.
3) A performance and computational cost trade-off analysis that offers guidance for deployment in latency-sensitive and interpretability-focused applications.

By quantifying these trade-offs, we aim to inform the selection of GNN architectures in practical fraud detection systems and provide a foundation for future hybrid or dynamic graph approaches.

The remainder of this paper is structured as follows. Section 2 reviews the background of GNN applications. Section 3 describes methodology and experimental setup. Section 4 presents the results and analysis. Section 5 concludes with key takeaways and future research directions.

II. BACKGROUND

Graph Neural Networks (GNNs) have emerged as a powerful deep learning paradigm for processing graph-structured data. Unlike traditional machine learning methods that operate on tabular data, GNNs use the connectivity and relationships within graphs to improve learning performance. In the context of fraud detection, financial transactions can be naturally represented as graphs, where nodes correspond to entities such as users or accounts, and edges represent interactions such as transactions. This structured representation enables GNNs to uncover complex patterns that may indicate fraudulent behavior.

A. Graph Neural Networks

GNNs extend conventional neural networks by incorporating graph structures through message passing and node aggregation mechanisms. Given a graph $G = (V, E)$, where V represents nodes and E represents edges, the general update rule in a GNNs is expressed as (1) where $h_v^{(l)}$ denotes the representation of node v at layer l, $\mathcal{N}(v)$ represents the set of neighboring nodes, $f(.)$ is an aggregation function that varies accross different GNN architectures, $W^{(l)}$ is a trainable weight matrix, and σ is a non-linear activation function.

Several variants have been proposed to improve standard GNN performance, including Graph Convolutional Networks (GCNs), Simplified Graph Convolutional Networks (SGCNs), GraphSAGE, Graph Attention Networks (GATs), and Graph Isomorphism Networks (GINs). GCNs employ a spectral-based

convolution operation that smooths node features across edges, defined as (2) where $H^{(l)}$is the node feature matrix at layer $l, \tilde{A} = A + I$ is the adjacency matrix with self-loops, and $\widetilde{D}$ is the diagonal degree matrix. In contrast, SGCNs remove the non-linearity (σ) and compress multiple graph convolutional layers into a single operation. Therefore, instead of stacking multiple layers, SGCNs apply one matrix multiplication with the precomputed propagation matrix. The SGCN update rule is simplified as (3).

GATs enhance message passing by introducing attention weights, allowing nodes to prioritize information from important neighbors as we see in (4). GraphSAGE extends GNNs through inductive learning, allowing the model to generalize to unseen nodes. Instead of relying on the full adjacency matrix, GraphSAGE samples a fixed number of neighbors and aggregates their features as we see in (5) where AGG can use mean aggregation, LSTM-based aggregation, or max pooling. GraphSAGE improves scalability, making it suitable for large-scale fraud detection in financial networks. Meanwhile, GINs strengthen the expressive power of GNNs by using a multi-layer perceptron (MLP) for feature transformation. The update rule is expressed as (6) where ϵ is a learnable parameter that controls the weight of self-information. GINs are particularly effective in distinguishing complex graph structures, making them useful for detecting subtle fraud patterns that other GNN models may overlook.

III. RESEARCH METHOD

A. Research Design

This study employs an experimental research design to systematically evaluate the effective-ness of different Graph Neural Network (GNN) models in detecting fraudulent transactions. The methodology consists of several key stages: data preprocessing, graph construction, model training, evaluation, and performance analysis. The primary objective is to assess the impact of different GNNs architectures on fraud detection performance while addressing the challenges posed by imbalanced datasets. The research follows a structured workflow.

1) Dataset Preprocessing: Raw financial transaction data undergoes cleaning, encoding, and normalization to ensure consistency. Fraudulent and legitimate transactions are labeled accordingly.
2) Graph Construction: Transaction networks are represented as graphs, where nodes correspond to entities, such as users or accounts, and edges represent transactions. Node fea-tures and edge attributes are extracted for meaningful representation.
3) Model Implementation: Five GNN models, GCNs, GraphSAGE, GATs, GINs, and SGCNs, are implemented using PyTorch Geometric. These models are trained and tested on both the original dataset and a synthetic oversampled dataset using SMOTE.
4) Evaluation and Analysis: Model performance is assessed using multiple fraud detection metrics, including F1-score, AUC-ROC, and precision-recall curves. The effect of class imbalance on model performance and the impact of SMOTE balancing techniques are also analyzed.

B. Datasets

The study uses four financial fraud datasets: Bank Fraud [20], IEEE-CIS [21], PaySim [22], and E-commerce [23]. Each dataset has a highly imbalanced class distribution, where fraudulent transactions are significantly outnumbered by legitimate ones. These datasets contain transactional details, including amount, timestamp, sender and receiver IDs, transaction type, and location. The dataset details are summarized in Table 1. Handling these imbalanced datasets is crucial because training models on skewed distributions may lead to biased predictions that favor the majority class.

1) Data Preprocessing and Cleaning

Before graph representations are constructed, the datasets undergo preprocessing and cleaning to ensure consistency and improve model performance. The following steps are applied:

- Missing Value Handling: Transactions with missing values in critical fields, such as amount, sender ID, and receiver ID, are either removed or imputed using statistical methods, such as mean or mode imputation.

TABLE 1
DATASETS DETAILS

Dataset	Total Transactions	Fraudulent Transactions	Legitimate Transactions	Fraud Ratio (%)
Bank Fraud	45211	5289	39922	11.70
E-commerce	1472952	73838	1399114	5.01
IEEE-CIS	10000	351	9649	3.51
PaySim	100000	20000	80000	20

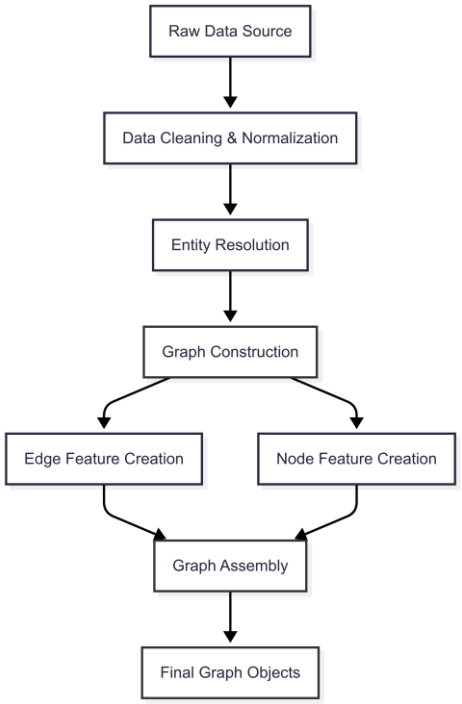

Figure 1. Graph Construction Process

$$x_{new} = x_i + \lambda\,(x_{NN} - x_i\,), \qquad \lambda \sim U\,(0,1) \tag{7}$$

- Feature Encoding: Categorical features, such as transaction type (e.g., transfer, cash-out, payment), are encoded using one-hot encoding or label encoding for compatibility with numerical models.
- Normalization: Continuous variables, such as transaction amount, frequency, and balance changes, are normalized using min-max scaling.
- Duplicate Removal: Identical transaction records are removed to prevent data leakage and overfitting.
- Graph Feature Extraction: The cleaned dataset is transformed into a graph-based structure by extracting node features, such as transaction history, account balance, and number of past frauds, and edge features, such as transaction amount, frequency, and transaction type.

2) Handling class imbalance

Fraud datasets are typically highly imbalanced, with fraudulent transactions making up less than 5% in some cases. This study implements the Synthetic Minority Over-sampling Technique (SMOTE) [24], [25] to oversample the minority fraud class in the node feature space. Specifically, SMOTE is applied after feature extraction and normalization but before GNN training, so it does not modify the original graph topology. This method generates synthetic fraudulent transactions to balance the dataset and prevent the models from overfitting to the majority class. SMOTE generates synthetic samples using the k-nearest neighbors (KNN) approach: for each minority-class instance x_i, identify its k-nearest neighbours in the minority class, randomly select a neighbor x_{NN}, and generate a synthetic sample x_{New}using linear interpolation.

Based on (7), λ is a random number sampled from a uniform distribution. By oversampling the minority class, SMOTE effectively mitigates class imbalance in this study. For comparison, the models' performance is also evaluated without SMOTE balancing.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + recall} \tag{9}$$

$$Precision = \frac{TP}{TP + FP}, Recall = \frac{TP}{TP + FN} \tag{10}$$

$$TPR = \frac{TP}{TP + FN}, FPR = \frac{FP}{FP + TN} \tag{11}$$

$$AUC = \sum_{i=1}^{n} (FPR_i - FPR_{i-1}) \times \frac{TPR_i + TPR_{i-1}}{2} \tag{12}$$

SMOTE was chosen over other balancing algorithms, such as random oversampling or ADASYN, because of its stability and its ability to generate synthetic samples that better preserve the original data distribution. This reduces the risk of overfitting and supports more consistent model performance.

3) Graph Construction

Figure 1 illustrates the graph construction process used in this study. Each dataset is represented as a transaction graph in which nodes correspond to user or account entities, and edges represent transactional interactions between them. The resulting graphs are modeled as directed graphs, reflecting the sender–receiver relationship inherent in financial transactions. Edge weights encode transaction-related attributes, such as transaction amount and interaction frequency, while unweighted edges are used when such information is unavailable.

Node features are derived through domain-specific aggregation of transactional records and include attributes such as transaction frequency, average transaction amount, account age, balance variation, and historical fraud count. Edge features capture transactional characteristics including amount, transaction type, and temporal occurrence. Before graph generation, all continuous features are normalized using min-max scaling, and categorical attributes are encoded using one-hot or label encoding, as appropriate.

The final graph is constructed using the PyTorch Geometric framework and represented as a tuple $G = (X, E, A)$, where X denotes the node feature matrix, E denotes the edge index and attributes, and A denotes the adjacency structure. No domain-specific pruning or edge augmentation is applied beyond the original transactional relationships, ensuring that the constructed graphs accurately reflect the underlying data distributions and can be reliably reproduced.

C. Evaluation Metrics

Four main evaluation metrics are used in this study: accuracy, F1-Score, and AUC-ROC. is defined as the ratio of correctly classified transactions, both fraudulent and legitimate, to the total number of transactions. Mathematically, accuracy is expressed as (8) where TP (True Positives) refers to correctly classified fraudulent transactions, TN (True Negatives) refers to correctly classified legitimate transactions, FP (False Positives) refers to legitimate transactions incorrectly classified as fraudulent, and FN (False Negatives) refers to fraudulent transactions incorrectly classified as legitimate.

Although accuracy provides an overall measure of classification performance, it may be unreliable in highly imbalanced datasets, such as fraudulent financial transaction datasets. For this reason, the F1-Score is also used because it balances precision and recall to provide a more meaningful evaluation in fraud detection tasks. The F1-Score is defined as (9) where (10).

Precision measures the proportion of predicted fraud cases that are actually fraudulent, whereas recall indicates the proportion of actual fraudulent transactions correctly identified by the GNNs. High precision means fewer false alarms, while high recall ensures that most fraud cases are detected.

Another important metrics for class imbalance is the Area Under the Receiver Operating Characteristics Curve (AUC-ROC). This metric evaluates the model's ability to distinguish between fraudulent

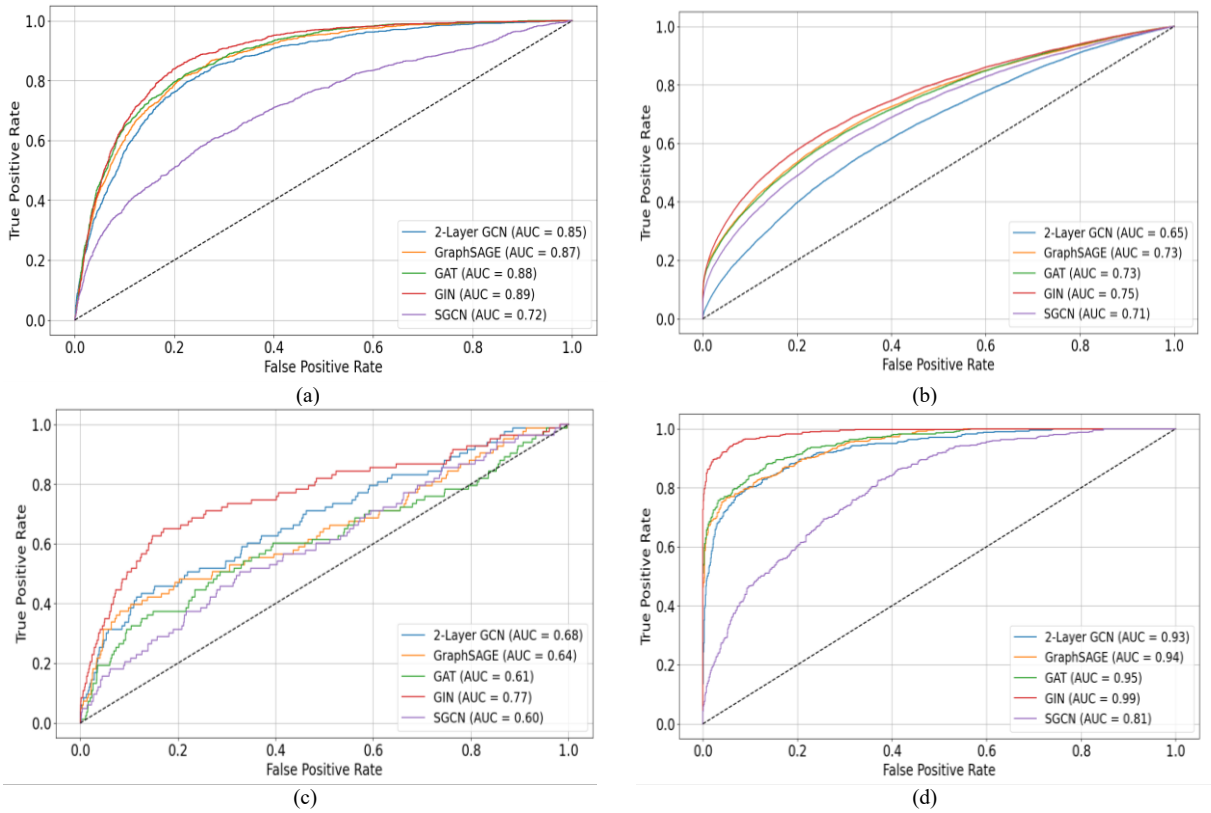

Figure 2. AUC Comparison of Each GNN Model Without SMOTE Balancing: (a) bank fraud; (b) e-commerce; (c) IEEE-CIS; and (d) PaySim.

and legitimate transactions across different classification thresholds. The ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR), where (11). AUC is computed as the sum of the areas under the ROC curve, often using the trapezoidal rule [26], which approximates it as (12). Equation (12) estimates AUC by summing the areas of trapezoids formed between consecutive points on the ROC curve. A higher AUC value, closer to 1, indicates better model performance in distinguishing fraudulent transactions from legitimate ones.

D. Experimental Setup

The experiments were conducted on a system equipped with an NVIDIA RTX 3060 GPU with 4 GB VRAM, an Intel Core i5 processor, 16 GB RAM, and running Ubuntu 22.04, while the software environment consisted of Python 3.13.2, PyTorch 2.6, and PyTorch Geometric 2.6.1. Five Graph Neural Network (GNN) models were implemented using PyTorch Geometric with a learning rate of 0.001, batch size of 512, 50 training epochs, Adam optimizer, hidden layer size of 64, and a dropout rate of 0.5. The experiments were executed on a high-performance computing environment utilizing an NVIDIA GPU to accelerate the training process. Furthermore, each model was trained using 80% of the dataset and evaluated on the remaining 20% unseen test set while preserving the original class distribution.

IV. Results and Discussion

This section discusses and analyzes the experimental results of the comparison of each GNN approach in detecting fraudulent transactions. Since the dataset is highly imbalanced, the experiment is divided into two conditions: with a balancing technique, namely SMOTE, and without balancing.

A. Experiment Without SMOTE

First, the performance comparison of each GNN method is illustrated in Figure 2. Figure 2 presents the AUC curve of GNN performance without SMOTE balancing. Across all datasets, GINs consistently achieve the highest AUC scores, particularly in the PaySim (0.98) and Bank Fraud (0.89) datasets. GraphSAGE and GATs also show strong and competitive performance, reinforcing their effectiveness

TABLE 2
ACCURACY AND F1-SCORE OF GNN MODELS WITHOUT SMOTE

	2-Layer GCN		GraphSAGE		GAT		GIN		SGCN	
Dataset	Accuracy	F1-Score	Accuracy	F1-Score	Accuracy	F1-Score	Accuracy	F1-Score	Accuracy	F1-Score
Bank Fraud	0.89	0.339	0.89	0.42	0.89	0.42	0.88	0.43	0.87	0.42
E-commerce	0.95	0.20	0.95	0.16	0.95	0.19	0.94	0.08	0.94	0.25
IEEE	0.96	0.22	0.96	0.22	0.96	0.22	0.96	0.18	0.94	0.22
PaySim	0.88	0.68	0.92	0.79	0.91	0.77	0.94	0.87	0.85	0.58

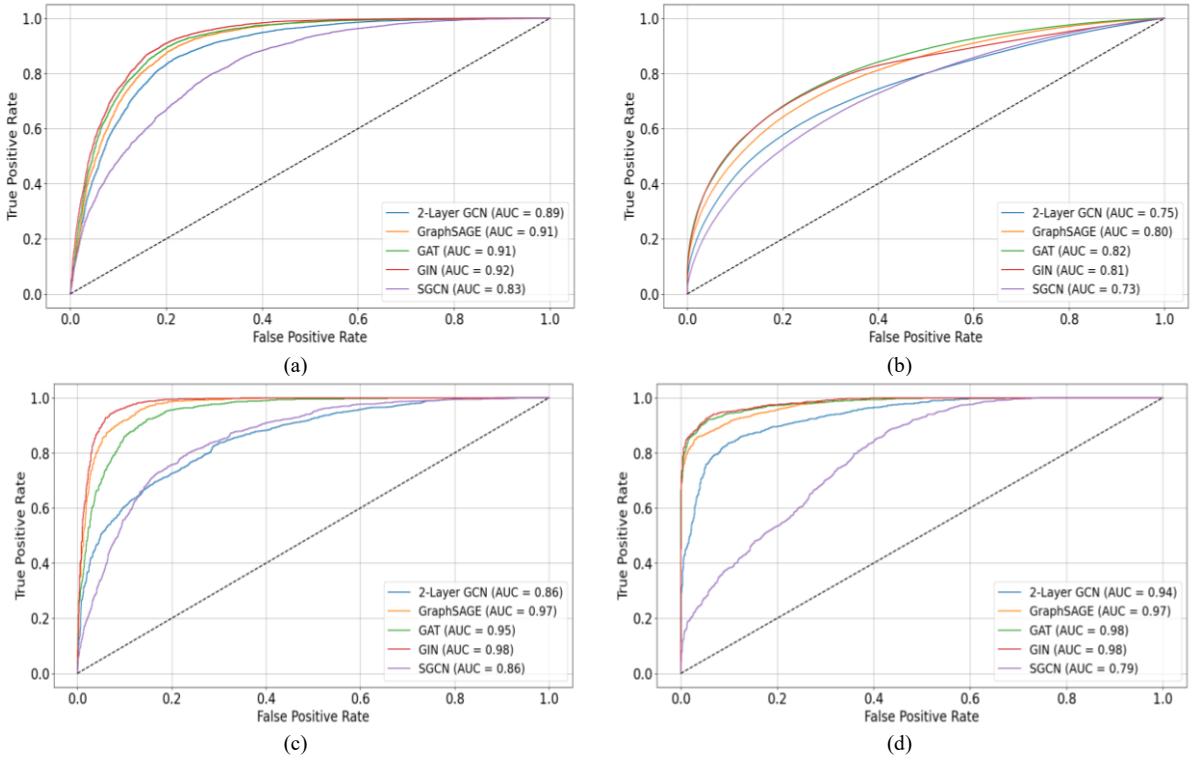

Figure 3. GNN Experiment Using SMOTE: (a) bank fraud; (b) e-commerce; (c) IEEE-CIS; and (d) PaySim.

in capturing complex fraud patterns. SGCNs, on the other hand, consistently underperform, indicating that their simplified convolutional approach is less effective.

Furthermore, Table 2 summarizes the performance of GNN models on different fraud detection datasets using accuracy and F1-score as key metrics. Across all datasets, accuracy remains consistently high, ranging from 0.94 to 0.96 in most cases, with minimal variation among models. However, F1-scores show significant differences, reflecting the challenges caused by class imbalance. On the Bank Fraud dataset, GraphSAGE, GATs, and GINs achieve the highest F1-scores (0.42–0.43), while GCNs performs notably lower at 0.339. In the E-commerce dataset, despite all models maintaining high accuracy (0.94–0.95), F1-scores vary widely, with SGCNs achieving the highest score (0.25) and GINs the lowest score (0.08). The IEEE-CIS dataset shows uniform accuracy (0.94–0.96), but with low F1-scores (0.18–0.22), indicating difficulty in fraud detection. In contrast, the PaySim dataset yields the highest F1-scores, with GINs performing best (0.87), followed by GraphSAGE (0.79) and GATs (0.77), while SGCNs underperform at 0.58.

B. Experiment With SMOTE

To reduce the impact of class imbalance and improve the predictive performance of Graph Neural Network (GNN) models in fraud detection tasks, the Synthetic Minority Over-sampling Technique (SMOTE) was applied to generate additional synthetic samples for the minority class. This approach aimed to improve the models' ability to correctly classify fraudulent transactions, which are often underrepresented in real-world datasets. Figure 3 illustrates the AUC score of each GNN model. The results indicate that each GNN model performs better with SMOTE than without SMOTE.

Among the tested models, GINs achieved the highest AUC scores, reaching 0.97 on the IEEE dataset and 0.98 on the PaySim dataset. GATs followed closely, with AUC scores of 0.91 on Bank Fraud and

TABLE 3
ACCURACY AND F1-SCORE OF GNN MODEL WITHOUT SMOTE

	2-Layer GCN		GraphSAGE		GAT		GIN		SGCN	
Dataset	Accuracy	F1-Score	Accuracy	F1-Score	Accuracy	F1-Score	Accuracy	F1-Score	Accuracy	F1-Score
Bank Fraud	0.84	0.84	0.84	0.85	0.84	0.85	0.85	0.85	0.80	0.81
E-commerce	0.73	0.72	0.74	0.73	0.74	0.73	0.74	0.73	0.72	0.71
IEEE-CIS	0.89	0.89	0.90	0.90	0.91	0.91	0.92	0.92	0.79	0.80
PaySim	0.87	0.87	0.90	0.90	0.91	0.91	0.93	0.92	0.79	0.78

TABLE 4
TIME EXECUTION ANALYSIS

Dataset	2-Layer-GCN	GraphSAGE	GAT	GIN	SGCN
Bank Fraud	1.07 ± 0.05	0.66 ± 0.03	1.9 ± 0.1	0.73 ± 0.04	0.64 ± 0.03
E-commerce	1.47 ± 0.07	0.34 ± 0.02	0.8 ± 0.05	0.3 ± 0.02	0.33 ± 0.02
IEEE-CIS	0.57 ± 0.03	0.27 ± 0.02	0.65 ± 0.03	0.26 ± 0.02	0.25 ± 0.02
PaySim	30 ± 1.5	23 ± 1.2	478 ± 20	66 ± 3	70 ± 3

0.94 on IEEE. GraphSAGE consistently performed well across datasets, with AUC scores ranging from 0.80 on E-commerce to 0.95 on IEEE. The 2-Layer GCN showed moderate performance, with AUC values between 0.78 on E-commerce and 0.94 on PaySim. SGCNs had the lowest AUC scores across all datasets, with values ranging from 0.75 on E-commerce to 0.86 on IEEE. Compared with the experiment without SMOTE, the AUC scores increased across all models and datasets. The most significant improvement was observed in the IEEE dataset, where GINs' AUC score increased from 0.79 to 0.97, and GraphSAGE's AUC in-creased from 0.77 to 0.95.

Furthermore, Table 3 presents the accuracy and F1-scores of various GNN models with SMOTE across four datasets. For the Bank Fraud dataset, GINs achieved the highest accuracy and F1-score, both at 0.85, while GraphSAGE and GATs followed closely with an F1-score of 0.85. The 2-Layer GCN model recorded an accuracy and F1-score of 0.84, whereas SGCNs had the lowest performance, with an accuracy of 0.80 and an F1-score of 0.81. In the E-commerce dataset, GraphSAGE, GATs, and GINs achieved the highest accuracy and F1-scores at 0.74 and 0.73, respectively. The 2-Layer GCN model recorded an accuracy of 0.73 and an F1-score of 0.72, while SGCNs had the lowest values, with an accuracy of 0.72 and an F1-score of 0.71.

C. Time Execution Analysis

The time analysis of different GNN models across multiple datasets reveals significant variations in computational efficiency. Table 4 shows the time needed to execute each GNN model. GraphSAGE and SGCNs consistently outperform other models in terms of speed, making them suitable for large-scale datasets. SGCN's efficiency comes from its simplified architecture, which removes unnecessary non-linearity and reduces computation time. GraphSAGE, by contrast, benefits from its neighbor-sampling mechanism, which avoids processing the full adjacency matrix as GCN does. Meanwhile, GINs show slightly higher execution times than GCNs because of their MLP-based aggregation, which offers improved expressiveness at a small computational cost. In contrast, GATs require the highest execution time, particularly on larger datasets, because of their pairwise attention computations. This trade-off makes GATs less favorable for large-scale graphs unless attention mechanisms are essential for the application.

Dataset size plays a major role in determining execution time, with PaySim showing significantly longer runtimes across all models because of its larger scale. The Bank Fraud and IEEE-CIS datasets are smaller and show much lower execution times, making them easier to process. The E-commerce dataset falls between these two conditions and shows moderate complexity. These findings suggest that GraphSAGE and SGCN are preferable choices for large datasets where efficiency is a priority, while GINs and GCNs serve as balanced alternatives when computational cost is not a major constraint. GATs, despite their expressiveness, remain the most computationally expensive and should be reserved for cases where attention-based mechanisms provide clear benefits.

D. Discussion

The findings of this study resonate with a growing body of recent research on GNNs in fraud detection and resource-constrained environments. For instance, the GAT-COBO framework [27] demonstrated the value of attention mechanisms in cost-sensitive fraud detection, although with an increased computational burden. Similarly, the heterogeneous GNN approach [28], which integrates graph attention and

a temporal decay mechanism with SMOTE and cost-sensitive learning, achieved notable improvements over benchmarks such as GCNs, GATs, and GraphSAGE on the IEEE-CIS dataset. These contemporary studies align with our observation that GINs and GATs offer expressive gains, particularly under imbalance and temporal complexity. However, our results contrast with findings in benchmark-based studies [8], [9], [10], [11], [12], which sometimes show minimal performance differences across simple GNN architectures under controlled conditions. By evaluating multiple GNNs across four distinct fraud datasets under both imbalanced and SMOTE-balanced conditions, this study fills a critical gap in understanding how architectural expressivity, imbalance handling, and real-world scale interact.

From an operational standpoint, GraphSAGE emerges as a highly appropriate model for deployment in latency-sensitive domains such as banking and e-commerce fraud detection. Its neighbor-sampling strategy enables efficient scaling to large graphs without substantially com-promising predictive performance, making it a practical choice for real-time inference pipelines. In contrast, GAT, while delivering strong discrimination and richer interpretability through its attention mechanism, requires substantially higher computational costs and is better suited for offline analysis or audit contexts where model explainability is prioritized over low-latency operation. These findings highlight a previously underexplored trade-off: practitioners often focus on standalone model performance rather than aligning GNN architecture with operational constraints. Our integrated analysis of performance metrics, execution time, and imbalance treatment addresses this oversight and offers actionable guidance for model selection in real-world fraud detection systems.

Specific architectural features and topologies consistently influenced model performance across datasets and imbalance ratios. Models with expressive aggregation, such as GIN with its MLP-based neighborhood encoding and GAT with attention-driven weighting, consistently improved recall for minority classes, particularly when paired with SMOTE. Conversely, GraphSAGE used its sampling mechanism to maintain a favorable balance between scalability and accuracy, especially in high-throughput scenarios. Meanwhile, the oversimplified design of SGCN, although computationally efficient, consistently underperformed in detecting rare fraud classes, showing that excessive architectural simplicity can reduce sensitivity. These patterns underscore a key implication: fraud-detection GNNs must be both architecturally expressive and operationally practical, tailored not only to accuracy metrics but also to graph topology, class imbalance, and deployment context.

V. CONCLUSION

To summarize, this study presents a comprehensive comparative analysis of five GNN models under both balanced and imbalanced settings across four fraud detection datasets, offering a performance and execution trade-off analysis. The results show that GraphSAGE achieves an optimal balance between scalability and accuracy, making it a strong candidate for latency-sensitive environments such as real-time banking and e-commerce fraud detection. In contrast, GIN provides superior representational power but at the cost of higher computational demand, making it more suitable for scenarios where accuracy is prioritized over execution speed. These findings provide actionable guidance for selecting GNN architectures in high-stakes financial contexts, where both detection performance and system responsiveness are critical.

Beyond the empirical comparisons, the findings of this study provide a foundation for several promising research directions. Although the current work focuses on static graph representations, the observed performance differences across GNN architectures highlight their varying suitability for extension into temporal and dynamic graph settings, where evolving transaction patterns and concept drift are critical considerations. In particular, models such as GraphSAGE and GAT may be adapted to incremental or streaming graph scenarios due to their sampling and attention mechanisms.

Additionally, although interpretability is not explicitly addressed in this study, the comparative results offer valuable insights for future work on explainable graph-based fraud detection, especially in regulatory-sensitive financial environments. Integrating attention visualization, sub-graph explanations, or post-hoc interpretability techniques could further enhance trust and accountability. By establishing a robust benchmarking baseline under both imbalanced and balanced conditions, this work serves as a stepping stone toward more adaptive, interpretable, and production-ready GNN-based fraud detection systems.

DECLARATION OF AI AND AI ASSISTED TECHNOLOGIES IN THE WRITING PROCESS

During the preparation of this work the authors used ChatGPT and Grammarly in order to improve the grammar of the article. After using this tool/service, the authors reviewed and edited the content as needed and takes full responsibility for the content of the publication.

CREDIT AUTHORSHIP CONTRIBUTION STATEMENT

Ricky Maulana Fajri: Conceptualization, Data Collection, Formal Analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, and Writing – review & editing. **Muhammad Gald Teary**: Conceptualization, Methodology, Writing – review & editing. **Ni Wayan Priscila Yuni Praditya**: Conceptualization, Methodology, Writing – review & editing.

DECLARATION OF COMPETING INTEREST

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

REFERENCES

[1] L. S. Goecks, A. L. Korzenowski, P. Gonçalves Terra Neto, D. L. de Souza, and T. Mareth, "Anti-money laundering and financial fraud detection: A systematic literature review," *Intell. Syst. Accounting, Financ. Manag.*, vol. 29, no. 2, pp. 71–85, 2022.

[2] P. Chatterjee, D. Das, and D. B. Rawat, "Digital twin for credit card fraud detection: Opportunities, challenges, and fraud detection advancements," *Futur. Gener. Comput. Syst.*, vol. 158, pp. 410–426, 2024.

[3] P. Vanini, S. Rossi, E. Zvizdic, and T. Domenig, "Online payment fraud: from anomaly detection to risk management," *Financ. Innov.*, vol. 9, no. 1, p. 66, 2023.

[4] A. O. Adewumi and A. A. Akinyelu, "A survey of machine-learning and nature-inspired based credit card fraud detection techniques," *Int. J. Syst. Assur. Eng. Manag.*, vol. 8, no. Suppl 2, pp. 937–953, 2017.

[5] J. O. Awoyemi, A. O. Adetunmbi, and S. A. Oluwadare, "Credit card fraud detection using machine learning techniques: A comparative analysis," in *2017 international conference on computing networking and informatics (ICCNI)*, IEEE, 2017, pp. 1–9.

[6] S. R. B. Reddy, P. Kanagala, P. Ravichandran, R. Pulimamidi, P. V Sivarambabu, and N. S. A. Polireddi, "Effective fraud detection in e-commerce: Leveraging machine learning and big data analytics," *Meas. Sensors*, vol. 33, p. 101138, 2024.

[7] R. Bin Sulaiman, V. Schetinin, and P. Sant, "Review of machine learning approach on credit card fraud detection," *Human-Centric Intell. Syst.*, vol. 2, no. 1, pp. 55–68, 2022.

[8] G. Zhen and L. Jianpin, "TGFFD: A Two-Stream Graph Neural Network for Financial Fraud Detection Based on Graph Convolution and Wavelet Analysis," in *2024 21st International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, IEEE, 2024, pp. 1–6.

[9] D. Wang *et al.*, "A semi-supervised graph attentive network for financial fraud detection," in *2019 IEEE international conference on data mining (ICDM)*, IEEE, 2019, pp. 598–607.

[10] D. Cheng, X. Wang, Y. Zhang, and L. Zhang, "Graph neural network for fraud detection via spatial-temporal attention," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3800–3813, 2020.

[11] M. M. Yadegar and H. Rahmani, "FinFD-GCN: Using Graph Convolutional Networks for Fraud Detection in Financial Data," *J. AI Data Min.*, vol. 12, no. 4, pp. 487–495, 2024.

[12] L. Lv, J. Cheng, N. Peng, M. Fan, D. Zhao, and J. Zhang, "Auto-encoder based graph convolutional networks for online financial anti-fraud," in *2019 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*, IEEE, 2019, pp. 1–6.

[13] A. Cherif, H. Ammar, M. Kalkatawi, S. Alshehri, and A. Imine, "Encoder–decoder graph neural network for credit card fraud detection," *J. King Saud Univ. Inf. Sci.*, vol. 36, no. 3, p. 102003, 2024.

[14] R. Li, Z. Liu, Y. Ma, D. Yang, and S. Sun, "Internet financial fraud detection based on graph learning," *IEEE Trans. Comput. Soc. Syst.*, vol. 10, no. 3, pp. 1394–1401, 2022.

[15] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv Prepr. arXiv1609.02907*, 2016.

[16] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv Prepr. arXiv1710.10903*, 2017.

[17] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[18] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," *arXiv Prepr. arXiv1810.00826*, 2018.

[19] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *International conference on machine learning*, Pmlr, 2019, pp. 6861–6871.

[20] I. Syamsuddin, M. A. Hanafie, and Z. Sharuna, "AIZBank : Bank Dataset for Fraud Prediction," Zenodo. [Online]. Available: https://zenodo.org/records/14636312

[21] V. Corporation, "IEEE-CIS Fraud Detection," Kaggle. [Online]. Available: https://www.kaggle.com/competitions/ieee-fraud-detection

[22] E. Lopez-Rojas, A. Elmir, and S. Axelsson, "PaySim: A financial mobile money simulator for fraud detection," in *28th European modeling and simulation symposium, EMSS, Larnaca*, Dime University of Genoa, 2016, pp. 249–255.

[23] S. Jagtap, "Fraudulent E-Commerce Transactions Dataset," Kaggle. [Online]. Available: https://www.kaggle.com/datasets/shriyashjagtap/fraudulent-e-commerce-transactions

[24] N. V Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.

[25] A. Sakho, E. Scornet, and E. Malherbe, "Theoretical and experimental study of SMOTE: limitations and comparisons of rebalancing strategies," *arXiv Prepr. arXiv2402.03819*, 2024.

[26] E. Süli and D. F. Mayers, *An introduction to numerical analysis*. Cambridge university press, 2003.

[27] X. Hu *et al.*, "GAT-COBO: Cost-sensitive graph neural network for telecom fraud detection," *IEEE Trans. Big Data*, vol. 10, no. 4, pp. 528–542, 2024.

[28] B. Wu, K.-M. Chao, and Y. Li, "Heterogeneous graph neural networks for fraud detection and explanation in supply chain finance," *Inf. Syst.*, vol. 121, p. 102335, 2024.