Scilight

*Transactions on Graph Intelligence and Network Applications*
https://www.sciltp.com/journals/tgina

*Article*

# Learning with Blacklists on Graphs via Prototype-Guided Dual-Frequency Filtering

Zhengyang Liu and Hang Yu *

School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China
* Correspondence: yuhang@shu.edu.com

**How To Cite**: Liu, Z.; Yu, H. Learning with Blacklists on Graphs via Prototype-Guided Dual-Frequency Filtering. *Transactions on Graph Intelligence and Network Applications* **2026**.

Received: 23 May 2026
Revised: 4 August 2026
Accepted: 4 August 2026
Published: 6 August 2026

**Abstract:** Graph-based fraud detection plays a critical role in identifying anomalous accounts and preventing financial losses in real-world systems, where graphs often contain millions of nodes but only a limited number of blacklist labels are available. Existing graph neural network approaches typically rely on full-graph message passing and sufficient supervision, which leads to weak neighborhood aggregation under scarce labels and poor scalability on large-scale graphs. In this work, we propose a scalable fraud detection framework that integrates prototype-guided graph abstraction with dual-frequency filtering to effectively learn from limited blacklists. Specifically, we construct class prototypes from the scarce labeled anomalies to capture compact class-level patterns and use them to guide subgraph abstraction and edge pruning, reducing the computational burden of graph message passing. On the resulting abstracted subgraphs, we apply dual-frequency graph filters to jointly model low-frequency homogeneous signals and high-frequency heterogeneous interactions. To further enhance robustness under incomplete supervision, we leverage high-confidence pseudo-labels to refine the graph abstraction and perform a second round of efficient dual-frequency filtering for prediction refinement. Extensive experiments on four benchmark datasets, including a large-scale graph, demonstrate that the proposed method consistently outperforms state-of-the-art approaches.

**Keywords:** fraud detection; graph neural networks; semi-supervised learning

## 1. Introduction

Fraud detection is a critical task aimed at preventing substantial financial losses caused by malicious activities. In recent years, graph-structured data has been increasingly applied to fraud detection due to its ability to naturally model complex relationships and interactions among entities [1–3]. This paradigm has shown great promise in various real-world scenarios, such as opinion fraud on online review platforms [4,5], illegal Bitcoin transactions [6], and money laundering [7,8]. Graph-based fraud detection formulates the task as node classification, where entities are nodes labeled as fraudulent or legitimate, and their interactions as edges.

Since suspicious accounts are detected by identifying irregular connection patterns within the graph, anomalous accounts may camouflage themselves by establishing links with legitimate nodes [9,10], as illustrated in Figure 1a. These connections, often referred to as heterogeneous edges (i.e., between nodes of different classes), undermine the homophily assumption commonly leveraged by standard Graph Neural Networks (GNNs) [11–13], thereby increasing the difficulty of accurate detection. To detect camouflaged anomalies, existing methods can be broadly categorized into three classes: (a) Neighborhood similarity sampling, which computes the similarity between neighboring nodes and the central node to selectively aggregate informative neighbors [14, 15]; (b) Frequency-aware aggregation, which utilizes differing frequency-domain signals to design multiple graph filters and enhance representation learning [7, 16, 17]; (c) Consistency-based regularization, which augments unlabeled data and leverages labeled nodes to enforce consistency constraints during training [18, 19].

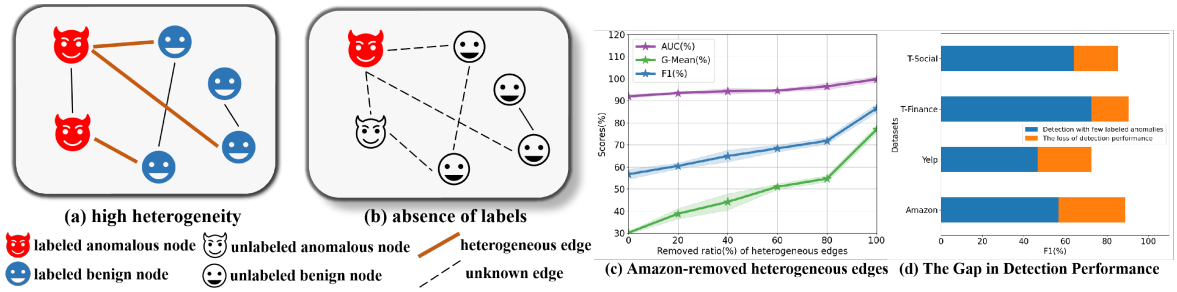


**Figure 1.** An illustration of challenges in graph-based fraud detection: (**a**) graph heterogeneity and (**b**) limited labeled anomalies; with (**c**) showing improved performance after removing heterogeneous edges; and (**d**) highlighting performance degradation under limited supervision.

Although these methods demonstrate good performance, they often assume access to sufficient labeled data from both classes [20–22]. However, in many real-world scenarios, only a few blacklist nodes based on past cases are available due to high cost of fraud investigation. This scenario suffers from the absence of labels for normal nodes, as shown in Figure 1b, leading to incomplete supervision and the lack of known homogeneous and heterogeneous information, ultimately diminishing accuracy of anomaly detection. As shown in Figure 1c,d, we conducted two experiments to demonstrate negative impact of graph heterogeneity and having only a small number of labeled anomalous nodes on graph-based fraud detection models. In subfigure (c), as the proportion of heterogeneous edges removed increases, detection performance improves progressively, even with just 10% of anomalous nodes labeled. In subfigure (d), the performance degeneration of PMP [17], which, with only 5% anomalous nodes labeled, experiences a performance drop of over 10% on four public datasets compared to models with both class labels in the same proportion.

Hence, we analyze the heterogeneity in graph-based fraud detection tasks and the decline in detection performance when only a few anomalous nodes are labeled. A semi-supervised framework that integrates prototype-guided graph abstraction with dual-frequency filtering is proposed. Our method first constructs class prototypes from the few labeled anomalous nodes to capture compact class-level patterns, which are then used to guide subgraph abstraction and edge pruning, effectively reducing graph heterogeneity and computational cost while preserving discriminative structural information. All unlabeled nodes are initially treated as normal, and neighborhood aggregation is performed using both low-pass and high-pass filters to capture homogeneous and heterogeneous signals. We then compute the similarity between each unlabeled node and the anomalous prototypes to estimate pseudo-labels, expanding the set of anomalous nodes for training. Based on these pseudo-labels, we further refine the graph by pruning a proportion of heterogeneous edges. Finally, the subgraph is fed into dual-frequency filters for a second round of aggregation, yielding refined and more accurate detection results under limited supervision.

The main contributions are summarized as follows:

- We analyze theoretically why only a small number of anomalous nodes are labeled and heterogeneity in graph would weaken detection model performance.
- We introduce a semi-supervised approach that enlarges the set of labeled nodes for training based on prototype learning, where the learned prototypes guide graph abstraction and edge pruning to reduce computational cost and graph heterogeneity. The method mitigates graph heterogeneity theoretically and improves detection performance.
- Experiments on real-world datasets demonstrate the superiority of our method in comparison with the state-of-the-art graph-based fraud detection methods.

The remainder of this paper is organized as follows: Section 2 reviews relevant research in the field. Section 3 defines the problem statement. Section 4 introduces design motivations and analysis. Section 5 describes our proposed methodology in detail. Section 6 presents the experimental results. Finally, Section 7 offers conclusions and discusses the implications of our findings.

## 2. Related Work

### *2.1. Graph-Based Fraud Detection*

GNNs represent nodes by aggregating their neighbors, making them highly effective for graph-structured data [23–27]. GeniePath [28] improved graph convolution by identifying significant receptive paths with adaptive path layers. GAS [29] enhanced embeddings by building homogeneous graphs that incorporate both local and global

context based on similarity. MAFI [30] utilized multiple aggregators and relationship-level attention to dynamically adjust significance of each relationship. Recent advancements in graph-based fraud detection have introduced several improvements. CARE-GNN [14] leveraged Multi-armed Bandits to adjust thresholds and select relevant neighbors. FRAUDRE [31] applied a balanced loss function and aggregated neighbors across multiple relations. PC-GNN [15] prioritized samples based on importance and oversampled fraudulent neighbors during aggregation. BWGNN [7] used multi-frequency filters with Beta kernels to capture high-frequency fraud features. GTAN [32] incorporated transaction labels for fraud detection, while AMNet [16] applied dual Bernstein polynomial filters with attention-based aggregation. GHRN [33] generated pseudo-labels from a pre-trained BWGNN model for further training. PMP [17] used specialized aggregation functions for different classes to better handle various types of neighbors. GAAP [34] detected fraud by first adaptively binning node attributes, combining them with neighbor-based association patterns via message passing to form local attribute–association motifs, and then aggregating these patterns globally to identify anomalies.

### *2.2. Graph Semi-Supervised Learning*

In practical scenarios, labeling large amounts of data is costly. Semi-supervised learning offers a solution by using unlabeled samples to boost model performance when labeled data is limited. Prior work [35] assumed feature similarity between neighboring nodes and utilized random negative sampling to create an unsupervised loss. Ref. [36] generated pseudo-labels from predictions for future epochs but failed to fully exploit connections between labeled and unlabeled samples. In recent years, there has been growing interest in settings with limited labeled data. Refs. [37,38] addressed challenges in highly limited supervision, suggesting improvements to pseudo-label quality and data categorization into super-classes to boost model performance. BSL [39] enhanced model detection by augmenting unlabeled nodes through consistency regularization, while ConsisGAD [18] tackled fraud detection with limited labeled data by augmenting node features with learnable mask functions for consistency. SpaceGNN [40] used a learnable projection to embed nodes into multiple geometric spaces, applied a distance-aware propagation based on weighted homogeneity, and ensembled across these spaces to improve detection. Unlike the challenges of the above approach, the real-world scenario we studied is more demanding. Only a small number of anomalous nodes is labeled in reality, making the above methods not suitable for direct application.

## 3. Problem Definition

An attributed graph involving fraudulent activities can be characterized as $G = \{V, \mathbf{X}, E, \mathbf{A}\}$. Here, $V = \{v_1, v_2, ..., v_N\}$ denotes the set of nodes, and $E = \{e_{ij}\}$ represents the set of edges, where each edge $e_{ij} = (v_i, v_j)$ signifies a connection between $v_i$ and $v_j$. Each node $v_i$ is associated with a $d$-dimensional attribute vector $\mathbf{x}_i \in \mathcal{R}^d$, which together form the feature matrix $\mathbf{X} \in \mathcal{R}^{N \times d}$. The adjacency matrix $\mathbf{A}$ is derived from the edges in $E$. Consequently, the task in this paper can be redefined as a node classification task, where each node is classified as either benign (label 0) or anomalous (label 1), i.e., $V \to \mathbf{Y} = \{0, 1\}^N$ with only a few nodes labeled as 1.

## 4. Design Motivations and Analysis

In this section, we first explain why only a small number of anomalous nodes are labeled leads to performance degradation. We then analyze the reasons why heterogeneity further exacerbates the performance degradation. Finally, we derive that it is feasible to estimate edge heterogeneity in the graph using aggregated features.

### *4.1. Why Having Only a Small Number of Labeled Anomalous Nodes Leads to Performance Degradation in Detection Models?*

In practical scenarios, only a small number of blacklisted nodes are labeled as training samples, with the remaining unlabeled samples treated as belonging to the benign class for supervised training. However, this approach results in a large amount of erroneous supervision, where many anomalous samples are misclassified as benign, confusing the feature spaces of both classes. On the other hand, in graph-structured data, node labels not only serve for direct node classification supervision but also influence the neighborhood feature aggregation. Whether labeled nodes are placed incorrectly into the wrong neighborhood aggregators or neighboring features are aggregated indiscriminately, it negatively impacts the distinguishability of features between the two classes of nodes.

### *4.2. Why Heterogeneity Exacerbates the Performance Degradation with Only a Few Anomalous Nodes Labeled?*

According to the theorem below, heterogeneity causes the distances between anomalous and benign nodes to become closer in the feature space, thereby making the task of identifying anomalous and benign nodes more challenging.

**Theorem 1.** *Let $\mathbf{h}^{(l)}$ be the embedding of node $\mathbf{x}^{(l)}$ after trained GNNs. Let $D_{PN}(\mathbf{x}) = \frac{1}{2}\sum \hat{\mathbf{A}}_{ij}||\mathbf{x_i} - \mathbf{x_j}||_2^2$ be a distance metric between anomalous and benign node embeddings $\mathbf{x}$ and $\hat{\mathbf{A}}_{ij}(y_i = 1, y_j = 0)$ is the normalized weight of heterogeneous edges. Then we have:*

$$D_{AB}(\mathbf{h}^{(l)}) \leq D_{AB}(\mathbf{x}^{(l)}) \tag{1}$$

**Proof.** Firstly, we can establish the relationship between $\mathbf{h}^{(l)}$ and $\mathbf{x}^{(l)}$ according to [41].

$$\begin{aligned} \mathbf{x}_i^{(l)} - \frac{\partial D_{AB}(x^{(l)})}{\partial \mathbf{x}_i^{(l)}} &= \mathbf{x}_i^{(l)} - \sum_{\mathbf{x}_j \in N(\mathbf{x}_i)} \hat{\mathbf{A}}_{ij}(\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}) \\ &= \sum_{\mathbf{x}_j \in N(\mathbf{x}_i)} \hat{\mathbf{A}}_{ij}\mathbf{x}_j^{(l)} = \mathbf{h}^{(l)} \end{aligned} \tag{2}$$

The Hessian of $D_{AB}$ is $\nabla^2 D_{AB}(\mathbf{x}) = I - D_{AB}^{-1}\mathbf{A}$. Since $D_{AB}^{-1}\mathbf{A}$ is Markov matrix, its eigenvalues are within $[-1, 1]$, so $2I - \nabla^2 D_{AB}(\mathbf{x}) = I + D_{AB}^{-1}\mathbf{A} \in [0, 2]$, i.e., $\nabla^2 D_{AB}(\mathbf{x}) \leq 2I$. Finally, we perform a second-Taylor expansion of $D_{AB}$ around $\mathbf{x}^{(l)}$ and obtain:

$$\begin{aligned} D_{AB}(\mathbf{h}^{(l)}) =& D_{AB}(\mathbf{x}^{(l)}) + \nabla D_{AB}(\mathbf{x}^{(l)})^T(\mathbf{h}^{(l)} - \mathbf{x}^{(l)}) + \\ & \frac{1}{2}(\mathbf{h}^{(l)} - \mathbf{x}^{(l)})^T \nabla^2 D_{AB}(\mathbf{x}^{(l)})(\mathbf{h}^{(l)} - \mathbf{x}^{(l)}) \\ =& D_{AB}(\mathbf{x}^{(l)}) - \nabla D_{AB}(\mathbf{x}^{(l)})^T \nabla D_{AB}(\mathbf{x}^{(l)}) + \\ & \frac{1}{2}\nabla D_{AB}(\mathbf{x}^{(l)})^T \nabla^2 D_{AB}(\mathbf{x}^{(l)}) \nabla D_{AB}(\mathbf{x}^{(l)}) \\ \leq& D_{AB}(\mathbf{x}^{(l)}) - \nabla D_{AB}(\mathbf{x}^{(l)})^T \nabla D_{AB}(\mathbf{x}^{(l)}) + \\ & \nabla D_{AB}(\mathbf{x}^{(l)})^T \nabla D_{AB}(\mathbf{x}^{(l)}) = D_{AB}(\mathbf{x}^{(l)}) \end{aligned} \tag{3}$$

□

Theorem 1 shows that heterogeneous connections reduce the separability between anomalous and benign nodes after message passing. Therefore, a natural strategy is to identify and suppress these harmful heterogeneous edges. This motivates the following analysis, which investigates why aggregated node features can be used to estimate edge heterogeneity.

### *4.3. Why Is It Feasible to Estimate Edge Heterogeneity with Aggregated Features?*

Since heterogeneity is high in graph-based fraud datasets, as shown in Figure 2, it becomes a natural idea to estimate the unlabeled edges so as to remove those that are highly heterogeneous. If label smoothing can be guaranteed by feature smoothing as stated in the following theorem with pseudo-labels that have relatively small errors, we can further leverage the neighborhood aggregation capability of the GNN to estimate node labels more accurately.

**Theorem 2.** *Suppose that the latent ground-truth mapping: $M : \mathbf{x} \to y$ from node features to node labels is differentiable and satisfies L-Libschitz constraint, i.e., $|M(\mathbf{x}_1) - M(\mathbf{x}_2)| \leq L||\mathbf{x}_1 - \mathbf{x}_2||$ for any $\mathbf{x}_1$ and $\mathbf{x}_2$ ($L$ is a constant). If the edge weights $a_{ij}$ approximately smooth $\mathbf{x}_i$ over its immediate neighbors with error $\epsilon_i$, i.e.,*

$$\mathbf{x}_i = \frac{1}{d_{ii}} \sum_{j \in N(i)} a_{ij}\mathbf{x}_j + \epsilon_i \tag{4}$$

*then edge weights $a_{ij}$ also approximately smooth $y_i$ over its immediate neighbors with the approximation error:*

$$\left| y_i - \frac{1}{d_{ii}} \sum_{j \in N(i)} a_{ij} y_j \right| \leq L||\epsilon_i||_2 + o\left( \max_{j \in N(i)} ||\mathbf{x}_j - \mathbf{x}_i||_2 \right) \tag{5}$$

**Proof.** Denote $\hat{a}_{ij} = a_{ij}/d_{ii}$ as the normalized weight of edge $(j, i)$. According to [42], a first-order Talyor expansion with Peano's form of remainder at $\mathbf{x}_i$ for $\sum_{j \in N(i)} \hat{a}_{ij} y_j$ is as following:

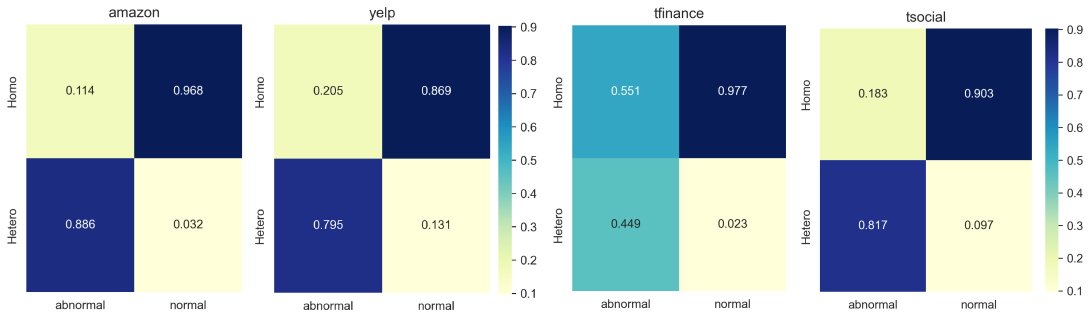


**Figure 2.** Statistics on the heterogeneity of the four graph-based fraud detection datasets.

$$\begin{aligned}
&\sum_{j\in N(i)} \hat{a}_{ij} y_j = \sum_{j\in N(i)} \hat{a}_{ij} M(\mathbf{x}_j) \\
&= \sum_{j\in N(i)} \hat{a}_{ij} (M(\mathbf{x}_i) + \frac{\partial M(\mathbf{x}_i)}{\partial \mathbf{x}^T}(\mathbf{x}_j - \mathbf{x}_i) + o(||\mathbf{x}_j - \mathbf{x}_i||_2)) \\
&= M(\mathbf{x}_i) + \frac{\partial M(\mathbf{x}_i)}{\partial \mathbf{x}^T} \sum_{j\in N(i)} \hat{a}_{ij}(\mathbf{x}_j - \mathbf{x}_i) + \sum_{j\in N(i)} \hat{a}_{ij} o(||\mathbf{x}_j - \mathbf{x}_i||_2) \\
&= y_i - \frac{\partial M(\mathbf{x}_i)}{\partial \mathbf{x}^T}\epsilon_i + \sum_{j\in N(i)} \hat{a}_{ij} o(||\mathbf{x}_j - \mathbf{x}_i||_2)
\end{aligned} \tag{6}$$

According to Cauchy-Schwartz inequality and $L$-Libschitz property:

$$|\frac{\partial M(\mathbf{x}_i)}{\partial \mathbf{x}^T}\epsilon_i| \leq ||\frac{\partial M(\mathbf{x}_i)}{\partial \mathbf{x}^T}||_2 ||\epsilon_i||_2 \leq L||\epsilon_i||_2 \tag{7}$$

Therefore, the approximation of $y_i$ is bounded by:

$$\begin{aligned}
&|y_i - \sum_{j\in N(i)} \hat{a}_{ij} y_j| \\
\leq &|\frac{\partial M(\mathbf{x}_i)}{\partial \mathbf{x}^T}\epsilon_i| + |\sum_{j\in N(i)} \hat{a}_{ij} o(||\mathbf{x}_j - \mathbf{x}_i||_2)| \\
\leq &L||\epsilon_i||_2 + o(\max_{j\in N(i)}(||\mathbf{x}_j - \mathbf{x}_i||_2))
\end{aligned} \tag{8}$$

□

## 5. Proposed Method

The overall framework of our method is illustrated in Figure 3, which mainly consists of three steps:

(a) Prototype-conditioned soft graph abstraction for scalability, which explicitly constructs a smaller, task-relevant computation subgraph for each mini-batch to improve scalability on large graphs.

(b) Unlabeled Node Supervision: For the labeled anomalous nodes, we use high-pass and low-pass graph filters to capture their high-frequency and low-frequency features, minimizing the likelihood of these nodes being classified as benign. At the same time, we use shared-parameter filters to encode the unlabeled nodes.

(c) Heterogeneous Edge Pruning with Enlarged Labeled Node Set: Using the prototype features constructed from labeled anomalous nodes, we assign pseudo-labels for more nodes and prune heterogeneous edges.

(d) Final Prediction Process: Utilizing the high-pass and low-pass filters from step (b) and the processed graph from step (c), we re-encode and classify nodes in the graph.

### *5.1. Prototype-Conditioned Graph Abstraction*

Although our model is trained in a mini-batch manner, the effective neighborhood size within each batch can still be prohibitively large on massive graphs, leading to excessive aggregation from nodes that are weakly related to the sparse blacklist supervision. To address this issue, we propose a prototype-conditioned graph abstraction mechanism that explicitly constructs a smaller yet task-relevant computation subgraph.

Let $G = (V, E)$ denote the original graph with node features $\mathbf{X}$. We first apply a lightweight linear projection to obtain preliminary node representations:

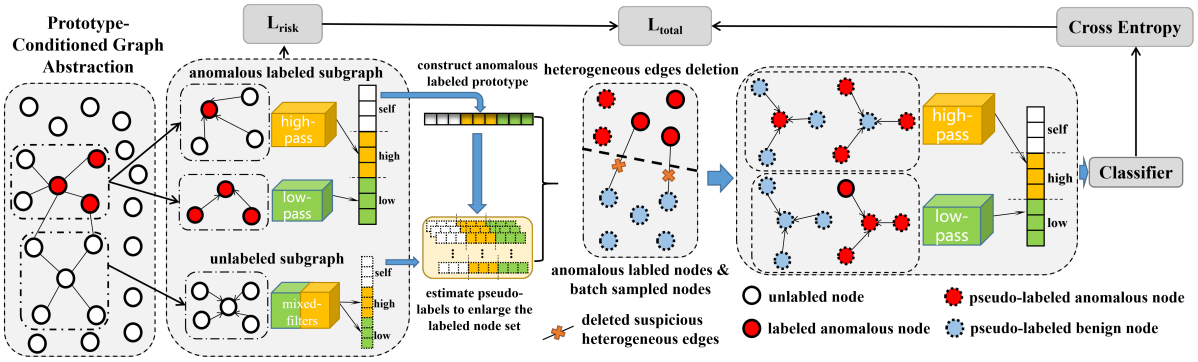


**Figure 3.** Overall workflow of the proposed framework. Prototype-conditioned graph abstraction first constructs task-relevant subgraphs for scalable computation. Dual-frequency filtering extracts complementary homogeneous and heterogeneous information. High-confidence pseudo-labels are then used to prune heterogeneous edges, followed by a second round of graph filtering for refined prediction.

$$\hat{\mathbf{X}} = \mathbf{X}\mathbf{W}_p, \tag{9}$$

where $\mathbf{W}_p$ is a learnable projection matrix.

Given a small set of labeled anomalous nodes $V_{\text{labeled}}$, we compute an anomalous class prototype as a semantic anchor:

$$\boldsymbol{\mu}_p = \frac{1}{|V_{\text{labeled}}|} \sum_{v_i \in V_{\text{labeled}}} \hat{\mathbf{x}}_i. \tag{10}$$

For each node $v_i$ appearing in a mini-batch and its neighbors, we compute its prototype similarity score:

$$a_i = \cos(\hat{\mathbf{x}}_i, \boldsymbol{\mu}_p). \tag{11}$$

Instead of preserving all neighbors, we retain both the top-$r\%$ most similar nodes and the top-$r\%$ most dissimilar nodes. The former are likely to share anomalous semantics with the prototype, while the latter may provide informative contrasting patterns that benefit boundary discrimination. Formally, the retained node set is constructed as

$$\tilde{V} = \hat{V}_{\text{sim}}^{\text{top-}r\%} \cup \hat{V}_{\text{dis}}^{\text{top-}r\%}, \tag{12}$$

where $\hat{V}_{\text{sim}}^{\text{top-}r\%}$ and $\hat{V}_{\text{dis}}^{\text{top-}r\%}$ denote the nodes with the highest and lowest prototype similarity scores, respectively.

We then construct the induced subgraph

$$\tilde{G} = G[\tilde{V}]. \tag{13}$$

This prototype-conditioned abstraction step removes a large number of less informative intermediate neighbors while preserving both prototype-aligned and prototype-opposite structural patterns, thereby substantially reducing the computational cost of message passing while maintaining discriminative context.

### *5.2. Unlabeled Node Supervision*

From the Laplace expression $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ defined on the abstracted graph $\tilde{G}$, we derive the frequency response function of GCN: $H_{GCN}(\lambda) = 1 - \lambda$, where $\lambda$ is eigenvalues of the graph Laplacian [43]. When $\lambda$ is small, GCN assigns a large amplitude to the signal. This indicates that GCNs tend to preserve low-frequency components, smoothing node features to enhance homogeneity. However, in graph-based fraud detection, the small proportion of anomalous nodes often connect to many benign nodes. These connections represent high-frequency signals, which are difficult for GCN to capture effectively.

Thus we propose to adopt dual graph filters. Specifically, a linear transformation is first employed to project the node attribute features $\mathbf{X}$ into a latent space $\hat{\mathbf{X}}$.

Referring to $H_{GCN}(\lambda) = 1 - \lambda$, we define frequency response functions of a high-pass filter $H_{\text{high}}(\lambda)$ and a low-pass filter $H_{\text{low}}(\lambda)$. $H_{\text{high}}(\lambda)$ strengthens as $\lambda$ increases, while $H_{\text{low}}(\lambda)$ strengthens as $\lambda$ decreases, shown as follows:

$$H_{\text{high}}(\lambda) = \lambda, H_{\text{low}}(\lambda) = 1 - \lambda \tag{14}$$

For labeled anomalous nodes, we roughly divide the adjacency matrix $\mathbf{A}$ into heterogeneous and homogeneous components —$\mathbf{A}_{\text{heter}}$ and $\mathbf{A}_{\text{homo}}$ (with unlabeled nodes initially assumed to be benign). This results in their respective eigenvalue matrices $\mathbf{\Lambda}_{\text{heter}}$, $\mathbf{\Lambda}_{\text{homo}}$. For heterogeneous and homogeneous neighborhoods, we apply the high-pass filter and low-pass filter, respectively:

$$\mathbf{Z}_{high}^{\text{labeled}} = F_{high}(\mathbf{\Lambda}_{\text{heter}}) = \mathbf{U}H_{high}(\mathbf{\Lambda}_{\text{heter}})\mathbf{U}^{\top}\hat{\mathbf{X}} \tag{15}$$

$$\mathbf{Z}_{low}^{\text{labeled}} = F_{\text{low}}(\mathbf{\Lambda}_{\text{homo}}) = \mathbf{U}H_{\text{low}}(\mathbf{\Lambda}_{\text{homo}})\mathbf{U}^{\top}\hat{\mathbf{X}} \tag{16}$$

For unlabeled nodes, directly aggregating features from their unlabeled neighbors may introduce noise. To address this, we extract neighborhood features using both high-pass and low-pass filters, and apply mix-up [44] to generate a feature space $\mathbf{Z}_{\text{unlabeled}}$.

$$\alpha = \text{sigmoid}(\mathbf{W}_{\alpha}\hat{\mathbf{X}} + \mathbf{b}_{\alpha}) \tag{17}$$

$$\mathbf{Z}_{\text{low}}^{\text{unlabeled}} = \alpha \cdot F_{\text{low}}(\mathbf{\Lambda}_{\text{unlabeled}}) \tag{18}$$

$$\mathbf{Z}_{\text{high}}^{\text{unlabeled}} = (1-\alpha) \cdot F_{\text{high}}(\mathbf{\Lambda}_{\text{unlabeled}}) \tag{19}$$

$$\mathbf{Z} = \text{Concat}(\hat{\mathbf{X}}, \mathbf{Z}_{\text{low}}, \mathbf{Z}_{\text{high}}) \tag{20}$$

where $\mathbf{W}_{\alpha}$ and $\mathbf{b}_{\alpha}$ are learnable parameters. $\mathbf{Z}$ is the overall feature matrix processed with dual graph filters.

A linear layer is applied to $\mathbf{Z}$ to obtain the prediction probability for each node.

$$p_i = \text{softmax}(\mathbf{W}_c\mathbf{z}_i + \mathbf{b}_c) = [p_i^0, p_i^1] \tag{21}$$

where $\mathbf{W}_c$ and $\mathbf{b}_c$ are learnable parameters for the classifier. $p_i^k$ is the predicted probability of node $v_i$ on class $k$.

Thus, we define the optimization objective in this step as minimizing the empirical risk of labeled anomalous nodes, i.e.,

$$\mathcal{L}_{risk} = \frac{1}{|V_{labeled}|} \sum_{v_i \ in V_{labeled}} \log(p_i^0) \tag{22}$$

### *5.3. Heterogeneous Edge Pruning with Enlarged Labeled Node Set*

To fully leverage the limited number of labeled anomalous nodes, we use the node prototypes constructed from $Z^{\text{labeled}}$ to assist in the estimation of pseudo-labels. First, we compute the mean of $Z^{\text{labeled}}$ to obtain the prototype center

$$\boldsymbol{\mu}_z = \frac{1}{|V_{\text{labeled}}|} \sum_{v_i \in V_{\text{labeled}}} \mathbf{z}_i \tag{23}$$

Thus, we compute the distance between the feature of each unlabeled node and the labeled anomalous prototype.

$$s_i = \|\boldsymbol{\mu}_z - \mathbf{z}_i\|_2 \tag{24}$$

$$\hat{V}_{anomaly} = \text{Sort}(s_i) \tag{25}$$

where $\text{s}_i$ is the distance between $\mathbf{z}_i$ and $\mu$, $\hat{V}_{anomaly}$ samples $r\%$ nodes sorted with $s_i$, with the subset having higher distance being estimated as potential anomalies. Since prototype matching is performed in the learned latent embedding space, Euclidean distance naturally measures the compactness around the anomaly prototype while maintaining computational efficiency.

The set of anomalous nodes is formed by the union of $\hat{V}_{\text{anomaly}}$ and initially labeled nodes $V_{\text{labeled}}$, with the remaining nodes considered benign. We assign pseudo-labels to all edges and remove the predicted heterogeneous edges, i.e.,

$$\hat{E} = E - E_{hetero} \tag{26}$$

where $\hat{E}$ is the pruned edge set. For any $e_{ij} \in E_{\text{hetero}}$, it satisfies that $v_i \in V_{\text{labeled}} \cup \hat{V}_{\text{anomaly}}$ and $v_j \in (V - V_{\text{labeled}} - \hat{V}_{\text{anomaly}})$.

The use of predicted pseudo-labels to assess the heterogeneity of edges introduces some error. However, we can demonstrate that under certain conditions, the impact of this error is minimal. We will first explain the scenario

where the distribution of first-order neighbor labels is used with true labels according to [45]. Since we used both high-pass and low-pass filters, we can obtain the neighbor node labels aggregated by the high-pass filter and the low-pass filter, denoted as $S^{high}$ and $S^{low}$.

$$S^{high} = \hat{L}Y, S^{low} = (I - \hat{L})Y \tag{27}$$

where $\hat{L}Y$ is the normalized Laplace matrix. Next, we consider $S$ in the presence of prediction error. Denote $\delta_i \geq 0$ as the prediction error of node $v_i$, we define the predicted label matrix $\hat{Y}$ under prediction error as:

$$\hat{Y}_i = \begin{cases} [1 - \delta_i, \delta_i], & \text{if } y_i = 0 \\ [\delta_i, 1 - \delta_i], & \text{if } y_i = 1 \end{cases} \tag{28}$$

Let $S_i^{low} = [S_{i0}^{low}, S_{i1}^{low}]$, and the expectation of prediction error for each class as $\mathcal{E}_k$, $k \in \{0, 1\}$. Therefore, if $y_i = 0$:

$$\begin{aligned} S_{i0}^{low} &= \frac{1}{d_i + 1}[(1 - \delta_i) + \sum_{j:y_i=y_j, i\neq j} (1 - \delta_j) + \sum_{k:y_i\neq y_j} \delta_j] \\ &= \frac{1}{d_i + 1}[(1 - \delta_i) + h_i(1 - \mathcal{E}_0) + (1 - h_i)\mathcal{E}_1] \\ &= \frac{1}{d_i + 1}[1 + (\mathcal{E}_1 - \delta_i) + h_i(1 - \mathcal{E}_0 - \mathcal{E}_1)] \end{aligned} \tag{29}$$

where $h_i$ indicates the where $h_i$ represents the proportion of homogeneous neighbor edges of node $v_i$. Similarly,

$$S_{i1}^{low} = \frac{1}{d_i + 1}[1 + (\delta_i - \mathcal{E}_1) + h_i(\mathcal{E}_0 + \mathcal{E}_1 - 1)] \tag{30}$$

In contrast, when $y_i = 1$,

$$S_{i0}^{low} = \frac{1}{d_i + 1}[1 + (\delta_i - \mathcal{E}_0) + h_i(\mathcal{E}_0 + \mathcal{E}_1 - 1)] \tag{31}$$

$$S_{i1}^{low} = \frac{1}{d_i + 1}[1 + (\mathcal{E}_0 - \delta_i) + h_i(1 - \mathcal{E}_0 - \mathcal{E}_1)] \tag{32}$$

for a qualified pseudo-labeling process, $\mathcal{E}_0 + \mathcal{E}_1 < 1$. What's more, if $\delta_i < \mathcal{E}_0$ and $\delta_i < \mathcal{E}_1$, we can derive that the low-pass filter encoding has some ability to identify heterogeneous edges, as demonstrated by:

$$\begin{aligned} &(S_i^{low} - \frac{1}{d_i + 1}) \cdot (S_j^{low} - \frac{1}{d_j + 1}) > 0 \\ &> (S_m^{low} - \frac{1}{d_m + 1}) \cdot (S_n^{low} - \frac{1}{d_n + 1}), y_i = y_j, y_m \neq y_n \end{aligned} \tag{33}$$

If the graph structure is dense enough, then $\frac{1}{d_i+1}$ can be ignored. According to Table 1, the average degrees of the four datasets are approximately 337, 104, 539, and 13, which generally satisfy the aforementioned condition. A similar conclusion can be derived using the high-pass filter that we adopted:

$$S_i^{high} \cdot S_j^{high} > 0 > S_m^{high} \cdot S_m^{high}, y_i = y_j, y_m \neq y_n \tag{34}$$

### *5.4. Final Prediction Process*

Then we use $\hat{E}$ to obtain the pruned adjacency matrix $\mathbf{A}_p$. By deriving the new eigenvalue matrix $\hat{\mathbf{\Lambda}}$, and following Equations (18)–(20), we obtain the final node features.

$$\hat{\mathbf{Z}}_{\text{low}} = \alpha \cdot F_{\text{low}}(\hat{\mathbf{\Lambda}}), \hat{\mathbf{Z}}_{\text{high}} = (1 - \alpha) \cdot F_{\text{high}}(\hat{\mathbf{\Lambda}}) \tag{35}$$

$$\hat{\mathbf{Z}} = \text{Concat}(\hat{\mathbf{X}}, \hat{\mathbf{Z}}_{\text{low}}, \hat{\mathbf{Z}}_{\text{high}}) \tag{36}$$

The same linear layer as Equation (21) is applied to $\hat{\mathbf{Z}}$. Thus, we can use cross-entropy to compute the supervised loss.

$$\hat{p}_i = \text{softmax}(\mathbf{W}_c\hat{\mathbf{z}}_i + \mathbf{b}_c) = [\hat{p}_i^0, \hat{p}_i^1] \tag{37}$$

**Table 1.** Graph anomaly detection datasets statistics.

| Dataset | Nodes | Features | Fraud (%) | Relation | Edges |
|---|---|---|---|---|---|
| Amazon | 11,944 | 25 | 6.87 | R-U-R<br>R-S-R<br>R-T-R | 49,315<br>3,402,743<br>573,616 |
| Yelp | 45,954 | 32 | 14.53 | U-P-U<br>U-S-U<br>U-V-U | 175,608<br>3,566,479<br>1,036,737 |
| T-Finance | 39,357 | 10 | 4.58 | - | 21,222,543 |
| T-Social | 5,781,065 | 10 | 3.01 | - | 73,105,508 |

$$\mathcal{L}_{CE} = \sum_i (\hat{y}_i log(\hat{p}_i^0) + (1 - \hat{y}_i) log(\hat{p}_i^1)) \tag{38}$$

where $\hat{y}_i = 1$ when $v_i \in V_{\text{labeled}} \cup \hat{V}_{\text{anomaly}}$, else $\hat{y}_i = 0$.

The total loss of the model is

$$\mathcal{L} = \mathcal{L}_{CE} + \theta \cdot \mathcal{L}_{risk} \tag{39}$$

where $\theta$ is a hypermeter to balance two loss terms. The overall framework of the proposed method is summarized in Algorithm 1.

### *5.5. Complexity Analysis*

Let $N = |V|$ and $M = |E|$ denote the numbers of nodes and edges in the original batch subgraph, respectively. For the prototype-conditioned abstracted graph $\tilde{G}$, the retained node and edge numbers are denoted as $\tilde{N} = |\tilde{V}|$ and $\tilde{M} = |\tilde{E}|$. The input and hidden feature dimensions are denoted as $d$ and $d'$, respectively.

#### 5.5.1. Prototype-Conditioned Graph Abstraction

The linear projection $\hat{\mathbf{X}} = \mathbf{X}\mathbf{W}_p$ requires $\mathcal{O}(N)$ time and $\mathcal{O}(Nd')$ space. Prototype similarity computation for retained nodes costs $\mathcal{O}(\tilde{N})$, while top-$r\%$ node selection requires $\mathcal{O}(\tilde{N} \log \tilde{N})$ time. Thus, the abstraction stage has complexity $\mathcal{O}(N + \tilde{N} + \tilde{N} \log \tilde{N})$, with space complexity $\mathcal{O}(Nd' + \tilde{N}d' + M + \tilde{M})$.

#### 5.5.2. Dual Graph Filtering and Edge Pruning

The proposed method avoids explicit eigendecomposition and performs high-pass/low-pass filtering using sparse message passing on $\tilde{G}$. Each filter requires $\mathcal{O}(\tilde{M})$ time and $\mathcal{O}(\tilde{N})$ space, resulting in overall dual-filter complexity $\mathcal{O}(2\tilde{M})$. For pseudo-label estimation, similarity computation and ranking require $\mathcal{O}(\tilde{N} + \tilde{N} \log \tilde{N})$, while heterogeneous edge pruning costs $\mathcal{O}(\tilde{M})$.

#### 5.5.3. Final Prediction Stage

After obtaining the pruned graph $\hat{G}$, the final dual-filter propagation and classification require $\mathcal{O}(|\hat{E}|)$ time and $\mathcal{O}(\tilde{N}d')$ space.

Overall, the total time complexity of the proposed framework is $\mathcal{O}(N + \tilde{M} + \tilde{N} \log \tilde{N} + |\hat{E}|)$, while the overall space complexity is $\mathcal{O}(Nd' + \tilde{N}d' + M + \tilde{M})$. Since $\tilde{N} \ll N$ and $\tilde{M} \ll M$ after prototype-conditioned abstraction, the dominant message-passing cost is performed on a much smaller subgraph.

## 6. Experiments

### *6.1. Experiment Setup*

In this section, we conducted comparative experiments on four real-world datasets with ten currently representative graph-based fraud detection methods to test the performance of our model. Additionally, we conducted ablation experiments to test the effectiveness of the designed steps. These contents were aimed at addressing the following questions:

Q1: Does our method outperform the state-of-the-art approaches with only a few anomalous nodes labeled?
Q2: How about the effectiveness of pruning heterogeneous edges?

**Algorithm 1** Overall Framework

**Require:** Original graph $G = (V, E)$, node features $\mathbf{X}$, labeled anomalous nodes $V_{\text{labeled}}$, batch size $B$, top-$r\%$ node selection ratio
**Ensure:** Node predictions $\hat{Y}$

**Precompute prototypes:**
$\hat{\mathbf{X}} \leftarrow \mathbf{X}\mathbf{W}_p$ ▷ linear projection
$\boldsymbol{\mu}_p \leftarrow \frac{1}{|V_{\text{labeled}}|} \sum_{v_i \in V_{\text{labeled}}} \hat{\mathbf{x}}_i$
**for** each mini-batch of nodes of size $B$ **do**
  **Prototype-Conditioned Graph Abstraction:**
  Compute cosine similarity $a_i = \cos(\hat{\mathbf{x}}_i, \boldsymbol{\mu}_p)$ for batch nodes and their neighbors
  Select the top-$r\%$ most similar nodes and the top-$r\%$ most dissimilar nodes
  Form the abstracted subgraph $\tilde{G} = (\tilde{V}, \tilde{E})$ from the selected nodes
  **Dual-Frequency Feature Extraction:**
  Compute Laplacian $\tilde{\mathbf{L}}$ of $\tilde{G}$
  Apply high-pass and low-pass filters to labeled nodes:
  $\mathbf{Z}_{high}^{\text{labeled}}, \mathbf{Z}_{low}^{\text{labeled}}$
  Apply dual filters with mix-up for unlabeled nodes:
  $\mathbf{Z}^{\text{unlabeled}} = \text{Concat}(\hat{\mathbf{X}}, \mathbf{Z}_{low}, \mathbf{Z}_{high})$
  **Pseudo-Label Assignment and Edge Pruning:**
  Compute prototype feature $\boldsymbol{\mu}_z = \text{mean}(\mathbf{Z}^{\text{labeled}})$
  Compute distances $s_i = \|\mathbf{z}_i - \boldsymbol{\mu}_z\|_2$ and select top-$r\%$ nodes as $\hat{V}_{\text{anomaly}}$
  Prune heterogeneous edges: $\hat{E} = E - E_{\text{hetero}}$
  **Final Prediction:**
  Recompute Laplacian $\hat{\mathbf{L}}$ for pruned graph $\hat{G} = (V, \hat{E})$
  Apply high-pass and low-pass filters on $\hat{G}$ to get $\hat{\mathbf{Z}}$
  Compute prediction probabilities $\hat{p}_i = \text{softmax}(\mathbf{W}_c \hat{\mathbf{z}}_i + \mathbf{b}_c)$
  Compute batch loss: $\mathcal{L}_{CE} + \theta \mathcal{L}_{risk}$
  Backpropagate and update model parameters
**end for**
**return** Node predictions $\hat{Y}$

Q3: Can the design of independent risk loss, dual graph filters, pruning heterogeneous edges lead to an improvement in anomaly detection performance?
Q4: Is our method sensitive to the key hyperparameters?
Q5: How does our method perform in the visualization of experimental results?

#### 6.1.1. Datasets

We use Amazon, Yelp [4], T-Finance, and T-Social [7] to evaluate the performance of our approach. The detailed statistics of these datasets can be found in Table 1. Specifically, Amazon and Yelp are multi-relational graph datasets, each containing three distinct types of relational edges. For these datasets, we independently aggregate features of each node based on each type of edge, and then concatenate the results to form the node's comprehensive feature representation.

- Amazon [4] is a dataset of fraud comments on musical instruments. Users are modeled as nodes and are connected according to the relation of common comments, similar comment content, and the same star rating. Users with more than 80% or less than 20% helpful votes are labeled as normal users and fraudsters, respectively.
- Yelp [4] is a dataset of fraud comments on restaurants and hotels. Comments are modeled as nodes and connected by three types of relationships: comments made by the same user, comments with the same rating for the same project, and comments given on the same project within a month. The labeling method is similar to Amazon.
- T-Finance [7] is a fraud transaction dataset released recently that regards accounts as nodes. The characteristics of accounts in the transaction network include registration date, login activity and transaction frequency. Nodes are linked if there has been a transaction between them. Accounts that have engaged in fraud activities such as money laundering and online gambling are labeled as fraud accounts by experts.
- T-Social [7] is a social fraud detection dataset recently released together with T-Finance and constructed in a similar way, except that the edges represent a friend relationship of more than three months between two account nodes.

#### 6.1.2. Compared Methods

We test our method by comparing it with ten up-to-date graph-based fraud detection models. Of these, GCN [46], GraphSAGE [47], GAT [48] are basic graph neural networks. AMNet [16] , BWGNN [7], GHRN [33], PMP [17], BSL [39], GAAP [34], and SpaceGNN [40] are recent qualified methods.

- GCN [46]: A fundamental graph neural network (GNN) that learns node features by aggregating information from neighboring nodes.
- GraphSAGE [47]: An inductive GNN that utilizes a sampling strategy to select a fixed number of neighbors, enabling batch-like training.
- GAT [48]: A basic GNN that employs an attention mechanism to aggregate information from neighboring nodes.
- AMNet [16]: A spectral GNN model employing dual filters to capture features in different frequency bands. These features are subsequently aggregated with an attention mechanism.
- BWGNN [7]: A Beta Wavelet GNN model designed to better capture features associated with fraud frequency bands through the use of multi-band filters.
- GHRN [33]: An approach that removes harmful heterogeneous connections in a graph by utilizing approximate labels obtained through pre-training with a qualified detection model.
- PMP [17]: A model that aggregates neighbors from different classes using distinct node-specific aggregation functions, addressing the challenges posed by heterophily-homophily mixed graphs in fraud detection.
- BSL [39]: A semi-supervised learning method inspired by consistency regularization, particularly effective in scenarios with extremely limited labeled samples.
- SpaceGNN [40]: A multi-space GNN framework that embeds nodes into diverse geometric spaces and propagates information via distance-aware mechanisms with limited labels.
- GAAP [34]: A fraud detection approach that aggregates local attribute–association motifs derived from adaptive attribute binning and neighborhood structure.

#### 6.1.3. Experiment Settings and Implementation

The learning rate is set as 0.01, and the weight decay rate is 1e-5. We set the embedding size to 64, and training epochs to 50. For all experiments, mini-batch training with batch size 128 is adopted, which ensures that the method is scalable and does not suffer from memory bottlenecks. 1%, 5%, and 10% anomalous nodes are regarded as labeled under three different settings. Our model is implemented in PyTorch 1.13.1 and PyG [49] with Python 3.9, while baselines are implemented with published codes on a NVIDIA RTX 4090 GPU. Unless otherwise specified, $\theta$ was set to 0.6 and the prototype selection ratio r was set to 4%.

#### 6.1.4. Evaluation Metrics

We evaluate the proposed method using the following metrics: AUC (Area Under the Curve) reflects the ability to distinguish between positive and negative classes. G-Mean (Geometric Mean) provides insight into the model's balance between true positive and true negative rates. F1-Macro computes F1 for each class individually and then averages them.

### *6.2. Overall Results (RQ1)*

We summarize the performance of our model and ten benchmark methods in Tables 2–4, when only 1%, 5%, and 10% of the anomalous nodes are labeled. $\pm$ in the tables represent the standard deviation of the results from five experiments conducted with different random seeds, and OOM denotes Out of Memory. It can be observed that our model achieves the best performance across almost all metrics under all conditions. Notably, the G-Mean scores of several early baselines are abnormally low. This is mainly because these methods fail to recall anomalous nodes, leading to extremely poor detection ability on fraud datasets with severe class imbalance. In contrast, our method maintains a much higher recall, resulting in a significantly better G-Mean. We also emphasize that all baselines are evaluated under the same experimental settings, ensuring a fair comparison. Among GCN, GraphSAGE, and GAT, only GraphSAGE comes close to the best detection performance on Amazon when 10% of the anomalous nodes are labeled, while in all other cases its performance still lags significantly behind the best results.

When it comes to the most recent five graph-based fraud detection methods, we observe that as the proportion of labeled anomalous nodes decreases, the performance degradation of these methods is more significant compared to our model. Since these methods do not specifically address the condition of single-class node labeling, they are forced to treat a large number of unlabeled nodes as normal nodes during training. This leads to a higher degree of coupling between the feature spaces of both classes, resulting in inferior performance compared to our model.

**Table 2.** Performance(%) of graph-based fraud detection on four datasets with only 1% anomalies labeled. The best results are shown in bold, the second-best results are indicated in parentheses.

| Methods \ Dataset | Amazon | | | Yelp | | | T-Finance | | | T-Social | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AUC | G-Mean | F1 | AUC | G-Mean | F1 | AUC | G-Mean | F1 | AUC | G-Mean | F1 |
| GCN (2016) | 86.83±0.57 | 19.57±1.52 | 52.05±0.55 | **(79.27)**±0.23 | 0.00±0.00 | 46.26±0.00 | 82.87±2.58 | 21.91±0.56 | 28.67±0.73 | 78.26±0.15 | 7.05±0.55 | 43.70±0.65 |
| GraphSAGE (2017) | 85.41±2.94 | 25.67±3.42 | 54.63±1.53 | 78.43±0.19 | 0.00±0.00 | 46.26±0.00 | 85.46±1.87 | 82.75±0.81 | 65.11±0.75 | 76.84±0.32 | 7.20±0.33 | 46.66±0.32 |
| GAT (2017) | 88.80±2.35 | 30.22±1.18 | 56.82±0.60 | 78.06±0.06 | 0.00±0.00 | 46.26±0.00 | 67.26±1.12 | 56.53±1.85 | 31.51±2.10 | 91.58±0.47 | 6.56±0.30 | 49.96±0.50 |
| AMNet (2022) | 88.66±1.57 | 51.40±1.63 | (69.54)±1.06 | 65.28±0.15 | 0.00±0.00 | 46.19±0.00 | 90.93±0.21 | 0.00±0.00 | 48.92±0.00 | OOM | OOM | OOM |
| BWGNN (2022) | 86.37±1.02 | 75.28±1.10 | 69.52±1.20 | 72.36±1.01 | 66.41±0.95 | 55.69±1.78 | 90.06±2.48 | 82.46±0.66 | 54.84±0.97 | 62.80±5.84 | 8.70±5.22 | 49.32±1.39 |
| GHRN (2023) | 84.62±0.89 | (77.95)±0.68 | 69.16±0.82 | 74.37±0.15 | (67.69)±0.30 | (57.75)±1.13 | 88.76±0.29 | 81.30±0.43 | 56.52±0.89 | 67.40±2.57 | 6.13±0.81 | 50.05±0.70 |
| BSL (2024) | 87.82±0.46 | 13.15±1.31 | 50.01±0.38 | 77.76±0.15 | 3.19±1.15 | 46.56±0.08 | 86.63±1.00 | 83.11±1.37 | 66.40±2.16 | 74.61±0.26 | 55.19±1.23 | 55.86±0.21 |
| PMP (2024) | (89.19)±1.43 | 21.89±2.21 | 52.92±0.91 | 75.93±1.70 | 0.00±0.00 | 46.26±0.00 | (95.20)±0.56 | 0.00±0.00 | 48.89±0.00 | (92.05)±0.15 | 9.86±6.79 | 50.52±1.48 |
| D-MVP (2025) | 83.25±2.76 | 20.19±3.96 | 51.19±2.02 | 75.16±0.68 | 0.00±0.00 | 46.26±0.00 | 86.39±0.52 | 80.17±1.15 | 65.29±0.94 | 80.64±0.35 | 62.31±3.87 | 58.17±2.03 |
| GAAP (2025) | 86.63±1.15 | 53.37±1.96 | 68.19±1.16 | 74.25±0.41 | 0.00±0.00 | 46.26±0.00 | 85.44±0.62 | 78.80±1.53 | 52.16±1.09 | 90.78±0.86 | 56.69±6.79 | 57.41±1.14 |
| SpaceGNN (2025) | 85.26±0.84 | 74.12±2.28 | 67.46±1.24 | 68.08±0.37 | 0.00±0.00 | 46.26±0.00 | 93.13±0.46 | (86.17)±1.53 | (75.68)±0.94 | 88.64±0.31 | (76.59)±2.37 | (65.83)±1.29 |
| Ours | **89.38**±1.27 | **78.22**±0.99 | **74.75**±0.83 | 78.49±0.75 | **68.92**±0.64 | **59.29**±1.46 | **95.69**±0.41 | **91.35**±0.28 | **86.01**±0.27 | **94.65**±1.01 | **85.61**±2.14 | **69.87**±2.22 |

**Table 3.** Performance(%) of graph-based fraud detection on four datasets with only 5% anomalies labeled. The best results are shown in bold, the second-best results are indicated in parentheses.

| Methods \ Dataset | Amazon | | | Yelp | | | T-Finance | | | T-Social | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AUC | G-Mean | F1 | AUC | G-Mean | F1 | AUC | G-Mean | F1 | AUC | G-Mean | F1 |
| GCN (2016) | 83.85±0.61 | 26.75±1.52 | 55.17±0.72 | 80.24±0.16 | 1.74±1.71 | 46.50±0.67 | 53.66±0.48 | 9.57±0.18 | 48.07±0.31 | 64.83±0.45 | 3.50±0.83 | 46.98±0.85 |
| GraphSAGE (2017) | 84.54±0.83 | 49.97±1.95 | 68.65±1.26 | 79.40±0.48 | 0.00±0.00 | 46.45±0.00 | 88.23±0.40 | 82.74±1.89 | 69.68±0.25 | 78.13±0.58 | 7.23±0.46 | 47.22±0.30 |
| GAT (2017) | 85.40±0.35 | 46.25±1.85 | 66.24±1.18 | 78.28±0.53 | 3.66±2.19 | 46.61±0.19 | 85.44±3.98 | 78.36±2.52 | 61.10±3.09 | 81.23±0.35 | 7.49±0.33 | 45.28±0.57 |
| AMNet (2022) | 89.66±1.40 | 69.12±2.19 | (81.20)±1.39 | 69.26±0.14 | 0.00±0.00 | 46.45±0.00 | 91.79±0.50 | 60.75±1.32 | 76.20±0.85 | OOM | OOM | OOM |
| BWGNN (2022) | 88.17±1.55 | (85.00)±2.66 | 74.53±0.50 | 76.85±0.51 | 64.95±0.70 | 59.84±1.02 | 91.72±0.27 | 86.26±0.27 | 71.31±1.16 | 72.88±3.10 | 50.79±1.81 | 56.51±0.54 |
| GHRN (2023) | 88.51±1.67 | 83.84±2.62 | 74.50±0.38 | 78.72±0.14 | (66.15)±0.37 | (61.51)±0.48 | 92.68±0.24 | 86.62±0.39 | 65.34±1.41 | 67.21±2.63 | 68.01±1.66 | 50.04±1.58 |
| BSL (2024) | 90.96±0.16 | 30.30±3.15 | 56.96±1.61 | 78.99±0.15 | 3.56±1.10 | 46.77±0.75 | 84.37±1.24 | 77.48±3.71 | 72.03±0.31 | 73.79±0.28 | 55.45±0.93 | 55.79±0.69 |
| PMP (2024) | (91.99)±0.76 | 30.15±0.56 | 56.68±2.22 | (81.30)±1.00 | 4.07±0.34 | 46.61±0.29 | (95.13)±0.35 | 54.95±4.07 | 72.43±2.67 | (95.37)±1.47 | (83.98)±2.62 | 64.19±3.04 |
| D-MVP (2025) | 84.93±0.84 | 50.26±1.81 | 68.06±1.23 | 78.56±0.63 | 0.00±0.00 | 46.45±0.85 | 87.74±1.06 | 81.75±1.46 | 66.82±0.58 | 84.49±0.83 | 72.16±2.73 | 63.29±1.26 |
| GAAP (2025) | 88.36±1.29 | 66.82±2.92 | 73.98±1.21 | 77.25±1.04 | 65.37±1.55 | 59.98±0.85 | 89.41±0.93 | 83.16±1.20 | 68.35±1.02 | 91.67±0.14 | 63.87±3.59 | 61.33±1.37 |
| SpaceGNN (2025) | 88.08±0.52 | 82.67±2.58 | 76.19±1.39 | 70.34±1.39 | 59.75±2.16 | 56.83±1.25 | 94.18±0.64 | (88.49)±1.68 | (77.36)±0.94 | 90.27±0.51 | 79.63±1.80 | (68.46)±0.73 |
| Ours | **92.64**±0.44 | **86.99**±0.46 | **82.92**±0.86 | **82.85**±0.79 | **69.08**±0.45 | **63.75**±0.32 | **96.02**±0.26 | **91.72**±0.38 | **86.27**±0.51 | **95.92**±0.20 | **88.07**±0.34 | **72.32**±0.49 |

**Table 4.** Performance(%) of graph-based fraud detection on four datasets with only 10% anomalies labeled. The best results are shown in bold, the second-best results are indicated in parentheses.

| Methods \ Dataset | Amazon | | | Yelp | | | T-Finance | | | T-Social | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AUC | G-Mean | F1 | AUC | G-Mean | F1 | AUC | G-Mean | F1 | AUC | G-Mean | F1 |
| GCN (2016) | 88.46±0.79 | 46.83±1.98 | 66.22±1.08 | 80.69±0.14 | 6.87±2.76 | 47.16±0.35 | 89.73±0.03 | 80.25±0.62 | 54.16±0.78 | 79.55±0.23 | 7.44±0.00 | 52.91±0.09 |
| GraphSAGE (2017) | (93.69)±0.64 | 69.91±0.56 | 81.51±0.33 | 78.07±1.70 | 3.07±2.25 | 46.76±0.17 | 89.58±0.16 | 86.79±1.10 | 74.80±0.40 | 74.41±0.26 | 4.74±0.08 | 55.05±0.17 |
| GAT (2017) | 93.67±0.53 | 57.26±0.97 | 73.57±0.63 | 78.52±0.12 | 9.65±2.11 | 47.26±0.43 | 90.08±0.18 | 85.66±0.87 | 66.76±0.33 | 74.32±5.73 | 3.17±0.02 | 49.41±0.06 |
| AMNet (2022) | 86.21±0.17 | 77.29±2.79 | (84.06)±0.40 | 69.24±0.61 | 0.00±0.00 | 46.63±0.00 | 91.79±0.60 | 74.38±2.18 | (82.81)±0.56 | OOM | OOM | OOM |
| BWGNN (2022) | 89.60±0.63 | 84.83±3.75 | 79.12±0.42 | 80.93±0.95 | (72.71)±0.26 | 62.08±0.92 | 93.57±0.58 | 87.70±0.37 | 74.00±1.52 | 74.24±1.10 | 68.55±0.66 | 50.96±0.86 |
| GHRN (2023) | 90.33±0.15 | (85.36)±3.81 | 79.02±0.65 | 79.01±0.18 | 70.75±0.72 | (63.43)±0.23 | 92.99±0.60 | (89.11)±0.19 | 74.52±0.66 | 74.55±0.53 | 72.43±0.91 | 49.34±0.33 |
| BSL (2024) | 91.68±0.12 | 57.72±4.03 | 75.82±1.13 | 83.82±0.21 | 23.11±2.31 | 52.10±0.92 | 85.81±0.22 | 79.30±3.19 | 76.40±0.62 | 73.48±0.05 | 55.17±0.57 | 56.73±0.61 |
| PMP (2024) | 92.45±0.14 | 44.31±5.36 | 64.30±3.05 | (84.18)±1.30 | 20.96±1.49 | 54.50±3.70 | 66.95±5.05 | 68.49±2.74 | 81.17±1.57 | (94.73)±0.32 | 65.98±2.89 | (73.41)±1.80 |
| D-MVP (2025) | 88.47±0.76 | 79.38±1.72 | 81.25±1.64 | 81.79±0.56 | 25.03±3.29 | 62.61±0.87 | 88.84±1.35 | 78.64±1.28 | 75.31±1.79 | 88.16±0.29 | 60.37±2.33 | 64.95±2.07 |
| GAAP (2025) | 90.45±0.42 | 80.70±2.72 | 75.39±0.76 | 78.58±0.73 | 70.21±1.62 | 63.18±1.11 | 93.00±2.15 | 87.92±2.05 | 82.57±0.77 | 92.05±0.15 | (79.86)±6.79 | 70.52±1.48 |
| SpaceGNN (2025) | 91.52±0.86 | 85.17±1.37 | 79.63±0.42 | 76.28±1.47 | 66.38±1.72 | 61.82±1.13 | (94.65)±0.48 | 88.20±1.36 | 81.41±0.62 | 93.26±0.51 | 75.04±2.89 | 67.94±0.83 |
| Ours | **94.04**±1.15 | **87.78**±0.50 | **84.60**±0.66 | **83.65**±0.57 | **69.03**±0.68 | **70.07**±0.70 | **95.28**±0.23 | **91.27**±0.63 | **87.77**±0.48 | **96.31**±0.10 | **88.68**±0.48 | **77.54**±0.31 |

### *6.3. Effectiveness Analysis of Pruning Predicted Heterogeneous Edges (RQ2)*

In Figure 4, we demonstrate changes in detection performance when removing heterogeneous edges with accurate edge class labels on the Yelp and T-Finance datasets. Additionally, we use a dashed line to indicate detection metrics achieved by our model with estimated heterogeneous edges removed. Our method achieves a performance comparable to accurately removing 60% of heterogeneous edges.

### *6.4. Ablation Study (RQ3)*

We conducted a series of ablation experiments with only 10% anomalous nodes labeled by excluding each design, as shown in Table 5, where *ours-prototype conditioned graph abstraction* does not refine the training mini-batch subgraphs; *ours-remove hetero* omits the heterogeneous edge pruning step; and *ours-independent loss* removes the empirical risk loss $\mathcal{L}_{risk}$ for labeled anomalous nodes. Note that we do not include a variant without pseudo-labels, since in that case the blacklist set cannot be expanded beyond the initially labeled nodes, which makes the task degenerate into a trivial supervised setting and defeats the purpose of blacklist expansion.

Our design has a positive impact across all datasets. In particular, on Amazon and T-Social, the pruning of heterogeneous edges significantly improved detection performance. This finding supports the inference that as long as the initial node classification is not severely misled, the estimated pseudo-labels used for heterogeneous edge removal can still lead to high accuracy. We further observe that the improvement brought by prototype-conditioned

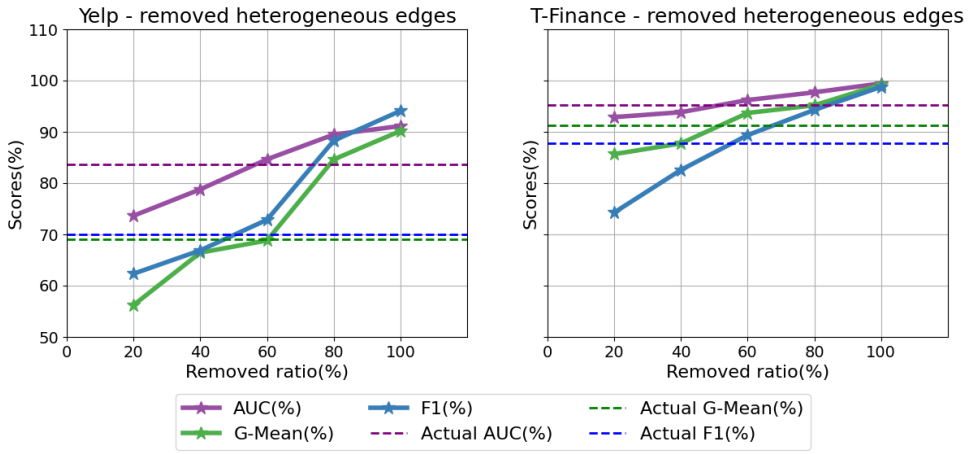


**Figure 4.** Performance of our method on Yelp, T-Finance under different heterogeneous-edge removal ratios.

**Table 5.** Ablation performance (%). The best results are shown in bold.

| Variant | Amazon | | | Yelp | | | T-Finance | | | T-Social | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AUC | G-Mean | F1 | AUC | G-Mean | F1 | AUC | G-Mean | F1 | AUC | G-Mean | F1 |
| Ours △ | 92.58 | 86.97 | 81.80 | 80.67 | 67.36 | 68.39 | 94.87 | 90.55 | 85.68 | 95.84 | 86.91 | 76.59 |
| Ours ♦ | 90.72 | 82.38 | 77.74 | 80.41 | 68.98 | 68.20 | 95.44 | 90.84 | 84.35 | 94.03 | 83.19 | 73.74 |
| Ours ∗ | 93.36 | 84.41 | 82.92 | 80.51 | 67.13 | 68.82 | 94.32 | 91.05 | 84.98 | 95.23 | 87.52 | 76.60 |
| Ours | **94.04** | **87.78** | **84.40** | **83.65** | **69.03** | **70.07** | **95.28** | **91.27** | **87.77** | **96.31** | **88.68** | **77.54** |

△: w/o prototype-conditioned graph abstraction; ♦: w/o heterogeneous edge pruning; ∗: independent loss; Ours: full model with all modules.

graph abstraction is more pronounced on Amazon and T-Social. This is because these datasets contain denser local neighborhoods and more redundant message-passing paths. By retaining only prototype-relevant nodes, the proposed abstraction effectively suppresses irrelevant aggregation while preserving discriminative structural information, leading to larger performance gains.

### *6.5. Sensitivity to Hyper-Parameters (RQ4)*

**The proportion $r\%$ of pseudo anomalous nodes.** According to the process of label estimation for unlabeled nodes, the rate $r\%$ based on the prototype similarity to labeled anomalous nodes is a key hyperparameter. Since the estimation involves some errors, balancing the recall of more anomalous nodes versus identifying a smaller but more accurate set of anomalous nodes becomes an important trade-off. **Weight factor analysis of the loss function.** As shown in Equation (39), the factor $\theta$ is critical in balancing $\mathcal{L}_{CE}$ and $\mathcal{L}_{risk}$. Specifically, we run our method on the Yelp dataset with 10% anomalous nodes labeled for $\theta \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$ and $r \in \{2, 4, 6, 8, 10\}$ based on grid search and report the results of all three evaluation metrics in Figure 5. Our method achieves better macro-F1 when $\theta$ and $r$ are around 0.6 and 4, respectively.

### *6.6. Visualization Analysis (RQ5)*

T-distributed stochastic neighbor embedding (t-SNE) [50] is a widely used dimensionality reduction technique that projects high-dimensional representations into a two-dimensional space, thereby facilitating intuitive visualization of feature distributions. To assess whether our model produces more discriminative representations, we apply t-SNE to the learned node embeddings on the Yelp and T-Finance datasets. As illustrated in Figure 6, the benign and anomalous nodes exhibit a clearer separation boundary after being mapped into the two-dimensional space. This distinct partition indicates that our model effectively enhances the inter-class separability while preserving the intra-class compactness of node features. Consequently, the visualization provides intuitive evidence that our method achieves the desired effect of distinguishing between different node categories.

## 7. Conclusions

In this paper, we addressed the challenges of graph-based fraud detection under scarce blacklist labels and large-scale, heterogeneous graphs. We proposed a scalable framework that integrates prototype-guided graph abstraction with dual-frequency filtering to effectively leverage limited labeled anomalies. Class prototypes capture compact anomaly patterns and guide subgraph abstraction, while high-confidence pseudo-labels enable

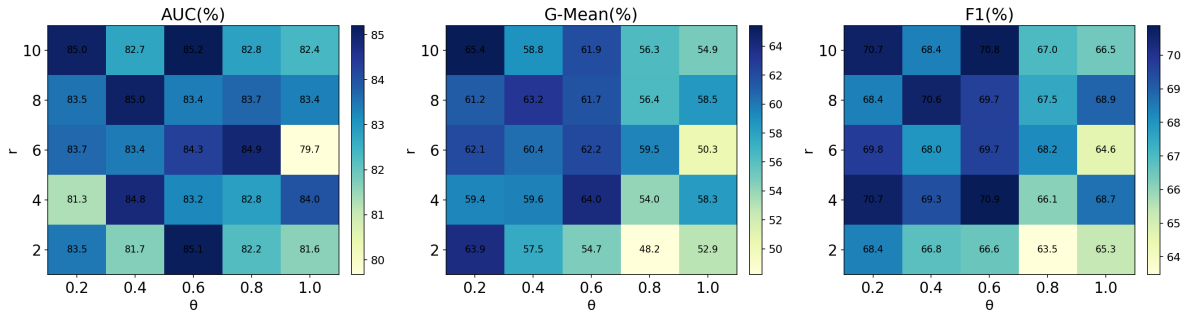


**Figure 5.** Experiments of sensitivity to hyper-parameters.

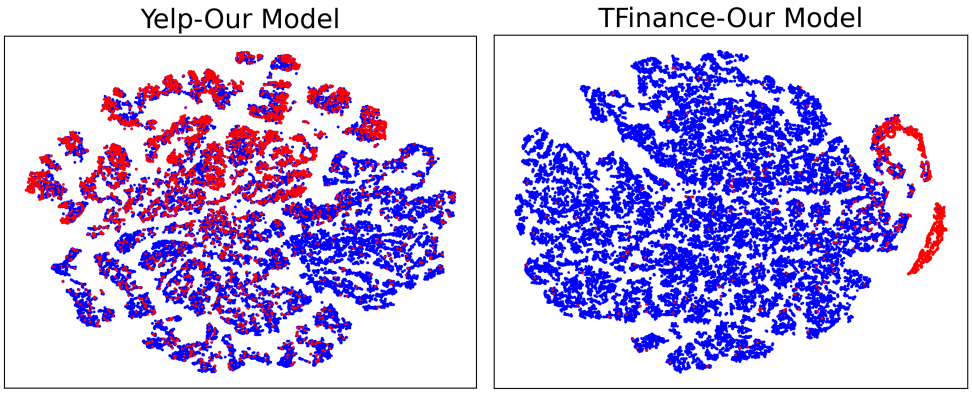


**Figure 6.** Visualization of the learned representations with TSNE on Yelp and T-Finance datasets.

selective edge pruning and further refine neighborhood aggregation. Extensive experiments on four real-world datasets, including a graph with nearly six million nodes, demonstrate that our method consistently outperforms state-of-the-art approaches under limited supervision, while achieving strong scalability and efficiency, highlighting its effectiveness for practical large-scale fraud detection. In future work, we will explore the proposed framework in more challenging settings, such as large-scale dynamic graphs and scenarios with more limited supervision. We will also investigate more efficient graph learning strategies to further improve the scalability of graph-based fraud detection.

## Author Contributions

Z.L.: conceptualization, methodology, software, writing—original draft preparation; H.Y.: supervision, validation, writing—reviewing and editing. Both authors have read and agreed to the published version of the manuscript.

## Funding

This work was supported by the National Natural Science Foundation of China Fund under Grant 62302287.

## Data Availability Statement

Not applicable.

## Conflicts of Interest

The authors declare no conflict of interest.

## Use of AI and AI-Assisted Technologies

No AI tools were utilized for this paper.

## References

1. Chen, J.; Chen, Q.; Jiang, F.; et al. SCN_GNN: A GNN-Based Fraud Detection Algorithm Combining Strong Node and Graph Topology Information. *Expert Syst. Appl.* **2024**, *237*, 121643.
2. Liu, Z.; Yu, H.; Luo, X. Mitigating Label Noise in Graph Learning with Information Bottleneck-Guided Aggregation and Adversarial Consistency. *Pattern Recognit.* **2026**, *180*, 114271.

3. Liu, Z.; Yu, H.; Luo, X. Federated Graph Anomaly Detection via Disentangled Representation Learning. In Proceedings of the ACM on Web Conference 2025, Sydney, NSW, Australia, 28 April–2 May 2025; pp. 1216–1224.
4. McAuley, J.J.; Leskovec, J. From Amateurs to Connoisseurs: Modeling the Evolution of User Expertise Through Online Reviews. In Proceedings of the 22nd International Conference on World Wide Web, Rio de Janeiro, Brazil, 13–17 May 2013; pp. 897–908.
5. Liu, Z.; Yu, H.; Luo, X. A Noise-Resistant Model for Graph-Based Fraud Detection. *Inf. Process. Manag.* **2025**, *62*, 104198.
6. Weber, M.; Domeniconi, G.; Chen, J.; et al. Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics. *arXiv* **2019**, arXiv:1908.02591.
7. Tang, J.; Li, J.; Gao, Z.; et al. Rethinking Graph Neural Networks for Anomaly Detection. In Proceedings of the 39th International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022; pp. 21076–21089.
8. Huang, L.; Liu, Z.; Luo, X.; et al. A Semi-supervised Approach with Dual Filters for Graph-Based Fraud Detection. In *Machine Learning and Knowledge Engineering for Decision Making*; Springer Nature Singapore: Singapore, 2027; pp. 37–50.
9. Liu, Z.; Dou, Y.; Yu, P.S.; et al. Alleviating the Inconsistency Problem of Applying Graph Neural Network to Fraud Detection. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Online, 25–30 July 2020; pp. 1569–1572.
10. Jin, D.; Liu, Z.; Li, W.; et al. Graph Convolutional Networks Meet Markov Random Fields: Semi-Supervised Community Detection in Attribute Networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 152–159.
11. Liu, S.; He, D.; Yu, Z.; et al. Beyond Homophily: Neighborhood Distribution-Guided Graph Convolutional Networks. *Expert Syst. Appl.* **2025**, *259*, 125274.
12. Shan, L.; Wang, N.; Zhao, J.; et al. Revisiting Positive Samples in Graph Contrastive Learning: From the Perspective of Message Passing. *arXiv* **2026**, arXiv:2606.10284.
13. Zhu, J.; Yan, Y.; Zhao, L.; et al. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 7793–7804.
14. Dou, Y.; Liu, Z.; Sun, L.; et al. Enhancing Graph Neural Network-Based Fraud Detectors Against Camouflaged Fraudsters. In Proceedings of the 29th ACM International Conference on Information and Knowledge Management, Online, 19–23 October 2020; pp. 315–324.
15. Liu, Y.; Ao, X.; Qin, Z.; et al. Pick and Choose: A GNN-Based Imbalanced Learning Approach for Fraud Detection. In Proceedings of the Web Conference 2021, Ljubljana, Slovenia, 19–23 April 2021; pp. 3168–3177.
16. Chai, Z.; You, S.; Yang, Y.; et al. Can Abnormality Be Detected by Graph Neural Networks? In Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22), Vienna, Austria, 23–29 July 2022; pp. 1945–1951.
17. Zhuo, W.; Liu, Z.; Hooi, B.; et al. Partitioning Message Passing for Graph Fraud Detection. In Proceedings of the Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024
18. Chen, N.; Liu, Z.; Hooi, B.; et al. Consistency Training with Learnable Data Augmentation for Graph Anomaly Detection with Limited Supervision. In Proceedings of the Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024
19. Liu, Z.; Yu, H.; Luo, X. A Semi-supervised Fraud Detection Model Based on Fuzzy Graph Neural Networks. *Appl. Soft Comput.* **2026**, *202*, 115913.
20. Liu, S.; He, D.; Yu, Z.; et al. Integrating Co-Training with Edge Discrimination to Enhance Graph Neural Networks Under Heterophily. In Proceedings of the AAAI Conference on Artificial Intelligence, Philadelphia, PA, USA, 25 February–4 March 2025; Volume 39, pp. 18960–18968.
21. Yu, Z.; Liang, C.; Chang, X.; et al. Dynamic Neighborhood Modeling via Node-Subgraph Contrastive Learning for Graph-Based Fraud Detection. In Proceedings of the AAAI Conference on Artificial Intelligence, Philadelphia, PA, USA, 25 February–4 March 2025; Volume 39, pp. 13115–13123.
22. Shao, Z.; Yu, H.; Wen, J.; et al. A Graph Fraud Detection Model Based on Mutual Information. *Neurocomputing* **2025**, *663*, 131972.
23. Yu, Z.; Jin, D.; He, D.; et al. Integrated Mixture of Neighborhood and Community Experts for Graph-Based Fraud Detection. In Proceedings of the ACM Web Conference 2026, Dubai, United Arab Emirates, 13–17 April 2026; pp. 604–613.
24. Liu, Z.; Gao, J.; Yu, H.; et al. A Robust Graph Fraud Detection Model Based on Adversarial Reweighting. *IEEE Trans. Comput. Soc. Syst.* **2025**, *12*, 5213–5224.
25. Yang, Q.; Yu, H.; Liu, Z.; et al. Synthesizing Global and Local Perspectives in Contrastive Learning for Graph Anomaly Detection. *Knowl.-Based Syst.* **2025**, *315*, 113289.
26. Liu, Z.; Gao, J.; Liao, Y.; et al. Knowledge-Enhanced Consistency Learning with Disentangled Multimodal Representation for Misinformation Detection. *Inf. Fusion* **2026**, *136*, 104501.
27. Liu, Z.; Yu, H.; Liao, Y.; et al. Fuzzy Federated Graph Learning via TSK Message Passing and Contrastive Uncertainty Learning. *IEEE Trans. Fuzzy Syst.* **2026**, 1–14. https://doi.org/10.1109/TFUZZ.2026.3713494.

28. Liu, Z.; Chen, C.; Li, L.; et al. GeniePath: Graph Neural Networks with Adaptive Receptive Paths. In Proceedings of the AAAI Conference on Artificial Intelligence; Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 4424–4431.
29. Li, A.; Qin, Z.; Liu, R.; et al. Spam Review Detection with Graph Convolutional Networks. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Honolulu, HI, USA, 27 January–1 February 2019; pp. 2703–2711.
30. Jiang, N.; Duan, F.; Chen, H.; et al. MAFI: GNN-Based Multiple Aggregators and Feature Interactions Network for Fraud Detection over Heterogeneous Graph. *IEEE Trans. Big Data* **2021**, *8*, 905–919.
31. Zhang, G.; Wu, J.; Yang, J.; et al. FraudRE: Fraud Detection Dual-Resistant to Graph Inconsistency and Imbalance. In Proceedings of the 2021 IEEE International Conference on Data Mining (ICDM), Auckland, New Zealand, 7–10 December 2021; pp. 867–876.
32. Xiang, S.; Zhu, M.; Cheng, D.; et al. Semi-Supervised Credit Card Fraud Detection via Attribute-Driven Graph Representation. In Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2023; pp. 14557–14565.
33. Gao, Y.; Wang, X.; He, X.; et al. Addressing Heterophily in Graph Anomaly Detection: A Perspective of Graph Spectrum. In Proceedings of the ACM Web Conference 2023, Austin, TX, USA, 30 April–4 May 2023; pp. 1528–1538.
34. Duan, M.; He, D.; Zheng, T.; et al. Global Attribute-Association Pattern Aggregation for Graph Fraud Detection. In Proceedings of the AAAI Conference on Artificial Intelligence, Philadelphia, PA, USA, 25 February–4 March 2025; Volume 39, pp. 11616–11624.
35. Wang, D.; Lin, J.; Cui, P.; et al. A Semi-Supervised Graph Attentive Network for Financial Fraud Detection. In Proceedings of the 2019 IEEE International Conference on Data Mining (ICDM), Beijing, China, 8–11 November 2019; pp. 598–607.
36. Kumar, A.; Ghosh, S.; Verma, J. Guided Self-Training Based Semi-Supervised Learning for Fraud Detection. In Proceedings of the Third ACM International Conference on AI in Finance, New York, NY, USA, 2–4 November 2022; pp. 148–155.
37. Lucas, T.; Weinzaepfel, P.; Rogez, G. Barely-Supervised Learning: Semi-Supervised Learning with Very Few Labeled Images. In Proceedings of the 36th AAAl Conferenceon Artificial Intelligence, Vancouver, BC, Canada, 22 February–1 March 2022; pp. 1881–1889.
38. Gui, G.; Zhao, Z.; Qi, L.; et al. Improving Barely Supervised Learning by Discriminating Unlabeled Samples with Super-Class. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 19849–19860.
39. Yu, H.; Liu, Z.; Luo, X. Barely Supervised Learning for Graph-Based Fraud Detection. In Proceedings of the 38th AAAl Conferenceon Artificial Intelligence, Vancouver, BC, Canada 20–27 February 2024; Volume 38, pp. 16548–16557.
40. Dong, X.; Zhang, X.; Chen, L.; et al. SpaceGNN: Multi-Space Graph Neural Network for Node Anomaly Detection with Extremely Limited Labels. *arXiv* **2025**, arXiv:2502.03201.
41. Wang, H.; Leskovec, J. Unifying Graph Convolutional Neural Networks and Label Propagation. *arXiv* **2020**, arXiv:2002.06755.
42. Wang, H.; Leskovec, J. Combining Graph Convolutional Neural Networks and Label Propagation. *ACM Trans. Inf. Syst.* **2021**, *40*, 1–27.
43. Shuman, D.I.; Narang, S.K.; Frossard, P.; et al. The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains. *IEEE Signal Process. Mag.* **2013**, *30*, 83–98.
44. Zhang, H.; Cisse, M.; Dauphin, Y. N.; et al. mixup: Beyond Empirical Risk Minimization. In Proceedings of the 6th International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
45. Luan, S.; Hua, C.; Lu, Q.; et al. Revisiting Heterophily for Graph Neural Networks. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 1362–1375.
46. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv* **2016**, arXiv:1609.02907.
47. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive Representation Learning on Large Graphs. In Proceedings of the 31st International Conference on Neural Information Processing System, Red Hook, NY, USA, 4–9 December 2017; pp. 1025–1035.
48. Veličković, P.; Cucurull, G.; Casanova, A.; et al. Graph Attention Networks. *arXiv* **2017**, arXiv:1710.10903.
49. Fey, M.; Lenssen, J.E. Fast Graph Representation Learning with PyTorch Geometric. *arXiv* **2019**, arXiv:1903.02428.
50. Van der Maaten, L.; Hinton, G. Visualizing Data Using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.