Decoupling Graph Neural Network with Contrastive Learning for Fraud Detection

Lin Meng[1], Yuxiang Ren[1], and Jiawei Zhang[2]

[1] IFM Lab, Department of Computer Science, Florida State University, FL, USA
[2] IFM Lab, Department of Computer Science, University of California, Davis, CA, USA
{lin,jiawei}@ifmlab.org, renyuxiang931028@gmail.com

Abstract. Recently, many fraud detection models introduced graph neural networks (GNNs) to improve the model performance. However, fraudsters often disguise themselves by camouflaging their features or relations. Due to the aggregation nature of GNNs, information from both input features and graph structure will be compressed for representation learning simultaneously. On the one hand, since not all neighbors provide useful information due to camouflage, aggregating information from all neighbors may potentially decrease the model performance. On the other hand, the structure including all neighbors is not reliable due to the relation camouflage. In this paper, we propose to decouple attribute learning and structure learning to avoid the mutual influence of feature and relation camouflage. Therefore, the model first learns its embedding seperately and then combine them together with label-guided contrastive losses to make predictions better. We conduct extensive experiments on two real-world datasets, and the results show the effectiveness of the proposed model.

Keywords: Fraud Detection · Camouflage · Graph Neural Networks

1 Introduction

Fraud detection is an important task in our daily life to fight against malicious actions or intentions. Fraudulent activities exist in many scenarios. For example, opinion fraud in online review platforms affects the behavior of customers [2], fake news in social media [1] misleads the opinion of people, and financial fraud in financial platforms can cause severe financial loss to customers [10, 17]. In these applications, the extensive interactions among people in both the online and offline world enable researchers to solve the problem from the graph perspective by treating all people as nodes and interactions as edges. Due to the superior representation power of graph neural networks (GNNs), GNN-based fraud detection has drawn extensive attention in both industry and academia [12, 11, 2]. Generally, GNNs aggregate all the information from the neighborhood and then update the information for the center node with it via a linear transformation [7, 21, 26]. Meanwhile, fraudsters camouflage themselves to avoid being detected by the fraud detection systems [5]. Typically, the camouflage behaviors of fraudsters can be categorized into attribute camouflage and relation camouflage. Attribute camouflage [2] refers to fraudsters fabricating their attributes like regular customers, while relation camouflage [2] indicates the relations that misguide the classifier.

The effectiveness of GNNs mainly comes from propagating information from a "homophily" neighborhood, which means the central nodes rely on information propagated from neighboring nodes in the same class. However, when dealing with fraud detection, the camouflage phenomenon violates the assumption since camouflage in GNNs produces noisy information propagated from nodes with different labels [2, 12]. Furthermore, attribute and connection are entangled during GNN propagation, making attribute learning and structure learning affect each other, which may exaggerate false information. The influence of connection camouflage can be illustrated by Fig. 1. Therefore, applying GNNs to fraud detection is challenging from three perspectives: (1) feature camouflage makes nodes with similar features have different labels; (2) connections camouflage leads GNN to aggregate noise information. More noisy information will be learned if with multiple GNN layers; and (3) both types of camouflage are mixed during the propagation at each layer, which further damages the performance of GNN.

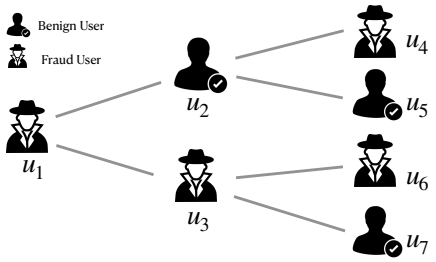


Fig. 1: Example of Introducing Noise When Applying A Two-layer GNN to Fraud Detection. In u_1-u_2-u_4, u_4 provides essential information for u_1, but it receives noise from u_2 unavoidably due to camouflage superposition. In u_1-u_3-u_7 and u_1-u_2-u_5, u_1 receives the noisy information from u_2, u_5 and u_7 due to camouflage on the first or second layer. Aggregated messages mislead GNN, making it predict wrong labels.

To address these aforementioned challenges posed by applying GNNs to fraud detection, we propose a framework that decouples attribute learning and structure learning, named **DC-GNN** (**D**ecoupled **C**ontrastive **G**raph **N**eural **N**etwork). To accomplish it, it contains three modules: (1) individual attribute encoding, which encodes four attributes together and then uses a graph transformer to identify useful neighbors that reveal characteristcs of fraudsters; (2) local structure encoding, which learns the structure feature that well identifies fraudsters; and (3) label-guided contrastive losses enhanced optimization, which contrasts the embeddings of fraudulent nodes and benign nodes so that the model learns more robust node embeddings.

In this paper, we detect fraudsters in multi-relational graphs. Existing models for multi-relational graphs mainly use the node embeddings learned by GNNs where they filter out camouflaged relations and then aggragate node features from "homophily" neighborhood [2, 12]. However, camouflaged relations can also reveal characteristics of fraudsters [16]. Thus, we use adjacency matrices to learn the structure characteristics of fraudster so that node features are not required during structure learning. The main contributions are summarized below:

– We propose a novel framework that learns attribute and structure embeddings separately to overcome the mutual effects brought by camouflage in GNNs for fraud detection.

- We propose a label-guided contrastive loss to enhance optimization, which improves the robustness of the model by contrasting fraudsters and benign nodes.
- We conduct extensive experiments on two real-world datasets, and the performance shows the effectiveness of the proposed model.

2 Notations and Problem Definition

Notations Generally, scalars are denoted as lowercase letters (e.g., x), the lowercase bold faced letters (e.g., $\mathbf{x}$) represent vectors, and capital bold faced letters (e.g., $\mathbf{X}$) to represent matrices. Sets or tensors are denoted as calligraphic letters (e.g., $\mathcal{X}$). $\mathbf{X}(i,:)$ and $\mathbf{X}(:,j)$ denotes i^{th} row and j^{th} column of $\mathbf{X}$, respectively. $\mathbf{X}(i,j)$ denotes the element in i^{th} row and j^{th} column of $\mathbf{X}$. $\|\cdot\|_F$ represents matrix F-norm. $\cup$ and $\odot$ represent concatenation and vector inner product, respectively.

Definition 1 *(Multi-relational Graph). A multi-relation graph can be represented as* $\mathcal{G} = (\mathcal{V}, \{\mathcal{E}_r\}|_{r=1}^{R}, \mathcal{X})$*, where* $\mathcal{V} = \{v_1, \ldots, v_n\}$ *denotes the node set,* $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ *represents a set of all node features. Each node* v_i *is associated with a* d*-dimensional feature vector* $\mathbf{x}_i \in \mathbb{R}^d$ *An edge* $e_{ij} = (v_i, v_j, r) \in \mathcal{E}_r$ *if* v_i *and* v_j *is connected via relation* $r \in \{1, \ldots, R\}$*. A label set* $\mathcal{Y}$ *denotes a node label set.*

In this paper, we relax the definition of terminology about "hop", which denotes the nodes within certain distance measures.

Definition 2 *(*K*-hop Neighbors) Suppose the distance between node* v_i *to* v_j *is* d_{ij}*. K-hop neighbors of node* v_i *are a set of nodes that* $d_{ij} \leq K$ *holds. Given a node* $v_i \in \mathcal{V}$ *in the graph, we use* $\mathcal{N}_i^{(K)}$ *to denote* K*-hop neighbors.*

Problem Definition. Given a multi-relational graph $\mathcal{G} = (\mathcal{V}, \{\mathcal{E}_r\}|_{r=1}^{R}, \mathcal{X})$ and the corresponding label set $\mathcal{Y}$ for all nodes (accounts or reviews) in $\mathcal{V}$, we model the fraud detection problem as a binary classification task. Formally, our goal is to find a function $f(\cdot)$, s.t.

$$f(\mathcal{G}) \rightarrow \mathcal{Y}.$$

where $\mathcal{Y} = \{0, 1\}_1^n$, and $y_i = 1$ if v_i is fraudulent while $y_i = 0$ if v_i is benign.

3 Proposed Method

To alleviate the effects of camouflage, we decouple the GNN learning process into individual attribute encoding and multi-relational local structure encoding; after that, we fuse attribute embedding and structure embedding as the final node representation. To better assist the learning process, we propose two label-guided contrastive losses. The overall framework is shown in Fig. 2.

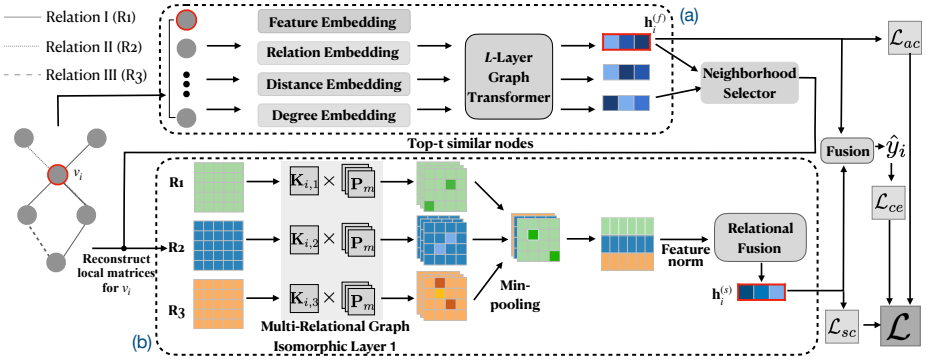


Fig. 2: DC-GNN Architecture. (Module (a): individual attribute encoding with a graph transformer; module (b): multi-relational local structure encoding with multi-relational IsoNN including multi-relational graph isomorphic layer, min-pooling layer, feature normalization (norm), and relational fusion; $\mathcal{L}$: the final loss, which includes $\mathcal{L}_{ce}$ for label prediction, $\mathcal{L}_{ac}$ and $\mathcal{L}_{sc}$ for attribute and structure contrastive losses, respectively.)

3.1 Individual Attribute Encoding

Node Attributes Since fraudulent nodes often camouflage themselves on node features, we propose to add three more graph related attributes to determine if a node is fraudulent or not. Since nodes are in a multi-relational graph, we choose connected relation types, distances towards the central node, and degrees under each relational graph. The reasons why we choose them as additional attributes are: (1) fraudsters are often involved in many social activities since they want to make profits. For example, in online review graphs, fraudsters often write an untruthful review for many items for better promotion. Therefore, fraudsters would connect to many users who also write reviews for those items; (2) additionally, many fraudsters are also involved in different types of activities, which potentially reveals the activity type of fraudsters, such as writing fake reviews or posting fake reviews at the same time; (3) moreover, the distances towards the given node show the influence of nodes towards the given node.

Formally, given a multi-relational graph $\mathcal{G} = (\mathcal{V}, \{\mathcal{E}_r\}|_{r=1}^{R}, \mathcal{X})$, for a node v_i, we encode the corresponding node feature $\mathbf{x}_i$ as follows

$$\mathbf{x}_i^{(a)} = \text{MLP}(\mathbf{x}_i) \in \mathbb{R}^{d_h \times 1}.$$

Since we are dealing with multi-relations, we use a vector $\mathbf{r}_i \in \{0,1\}^R$, where $\mathbf{r}_i(j) = 1$ indicates the node has a j-th relation type connection. One MLP layer is adopted to get the embedding of the relation vector.

$$\mathbf{x}_i^{(r)} = \text{MLP}\left(\mathbf{r}_i\right) \in \mathbb{R}^{d_h \times 1}.$$

We use the postional embedding proposed in [20] to encode the distance $P(v_i)$ of v_i to a node (i.e., the distance is decided by the shortest distance between v_i to the central

node in this paper)

$$\begin{aligned}\mathbf{x}_i^{(d)} &= \text{Pos-emb}(\text{P}(v_i)) \\ &= \left[\sin\left(\frac{\text{P}(v_i)}{10000^{\frac{2l}{d_h}}}\right), \cos\left(\frac{\text{P}(v_i)}{10000^{\frac{2l+1}{d_h}}}\right)\right]_{l=0}^{\left\lfloor\frac{d_h}{2}\right\rfloor},\end{aligned}$$

where $\mathbf{x}_i^{(d)} \in \mathbb{R}^{d_h \times 1}$. The index l iterates throughout all the entries in the above vector to compute the entry values with $sin(\cdot)$ and $cos(\cdot)$ functions for the node based on its distance. For degree d_i, since its number could be extremely large, we first normalize it and use MLP to obtain the information of it.

$$\mathbf{x}_i^{(e)} = \text{MLP}\left(\frac{d_i}{\sum_{r=1}^{R}\|\mathcal{E}_r\|}\right) \in \mathbb{R}^{d_h \times 1}.$$

By adding four attributes together, we obtain the initial node embeddings for all nodes.

$$\mathbf{x}_i' = \mathbf{x}_i^{(a)} + \mathbf{x}_i^{(r)} + \mathbf{x}_i^{(d)} + \mathbf{x}_i^{(e)}.$$

Attribute Encoding with Graph Transformer Due to the great representation power of transformer [20], we use it as our attribute encoder. In graphs, we regard the context of a node as its neighborhood. Therefore, we choose K-hop neighbors directly as the context of the given node.

For each node, we obtain its K-hop neighbors by preprocessing the graph data. Suppose we have the K-hop neighbor set $\mathcal{N}_i^{(K)}$ for v_i, the corresponding input feature matrix $\mathbf{X}_i \in \mathbf{R}^{(|\mathcal{N}_i^{(K)}|+1)\times h}$ and $\mathbf{X}_i = [\mathbf{x}_i', \mathbf{x}_{i,1}', \cdots, \mathbf{x}_{i,|\mathcal{N}_i^{(K)}|}']$. We set $\mathbf{H}_i^{(0)} = \mathbf{X}_i$ and feed it into the L-layer transformer:

$$\begin{aligned}\mathbf{H}_i^{(l)} &= \text{Transformer}\left(\mathbf{H}_i^{(l-1)}\right) \\ &= \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_h}}\right)\mathbf{V}\end{aligned}, \text{where} \begin{cases}\mathbf{Q} = \mathbf{H}_i^{(l-1)}\mathbf{W}_Q^{(l)} \\ \mathbf{K} = \mathbf{H}_i^{(l-1)}\mathbf{W}_K^{(l)} \\ \mathbf{V} = \mathbf{H}_i^{(l-1)}\mathbf{W}_V^{(l)}\end{cases}$$

where $\mathbf{H}_i^{(L)} \in \mathbb{R}^{(|\mathcal{N}_i^{(K)}|+1)\times d_h}$, and $\mathbf{W}_Q^{(l)}, \mathbf{W}_K^{(l)}, \mathbf{W}_V^{(l)} \in \mathbb{R}^{d_h \times d_h}$. Therefore, we get the embeddings of $\mathbf{H}_i^{(L)} = [\mathbf{h}_i^{(L)}, \mathbf{h}_{i,1}^{(L)}, \cdots, \mathbf{h}_{i,|\mathcal{N}_i^{(K)}|}^{(L)}]$. Note that due to the feature camouflage, we take the $\mathbf{h}_i^{(L)}$ directly as the final attribute embedding $\mathbf{h}_i^{(f)}$ of node v_i to avoid incorporating embeddings from the benign class.

3.2 Multi-relational Local Structure Encoding

Relations between nodes also reveal important information for detecting fraudsters. As indicated in previous works [16], fraudsters often act collectively, which also means the fraudsters may share similar local structures. However, traditional GNN cannot learn good structural information since fraudsters often hide themselves in connections with benign nodes, which introduce features belongs to benign nodes. To better learn the

local structural features, we propose to encode the adjacency matrix solely. Thus, we utilize isomorphic graph neural network [14]. However, IsoNN originally is used for homogeneous graph classification, we extend it to multi-relational graphs to capture the local structure embeddings for each node.

Local Adjacency Matrices Reconstruction. We need to reconstruct the local adjacency matrices for all nodes and then each node learns its local structure representation. However, reconstructing the local adjacency matrices with its K-hop neighborhood is not realistic, since it may involve several nodes, causing high computational costs. Therefore, we select useful nodes from K-hop neighbors. Nodes are selected based on the similarities between embeddings in the output of transformer $\mathbf{H}_i^{(L)}$ of the given node and its neighbors. Here, we calculate the similarity as follows:

$$s_{ij} = \mathrm{sim}(\mathbf{h}_i, \mathbf{h}_{i,j}) \ \ \forall v_j \in \mathcal{N}_i^{(K)}$$

Here, Euclidean distance is the similarity measurement. We select Top-t similar nodes $\{v_j, \cdots, v_k\}$ and itself v_i to reconstruct local adjacency matrix. Note that for the rest of the paper, we treat $\{v_j, \cdots, v_k\}$ together with central node v_i as selected t nodes just for simplicity.

With the multi-relational graph $\mathcal{G} = (\mathcal{V}, \{\mathcal{E}_r\}|_{r=1}^{R}, \mathcal{X})$, and the selected nodes $\{v_i, v_j, \cdots, v_k\}$, we reconstruct local adjacency matrices for all relations, which is denoted as $\mathcal{A}_i = \{\mathbf{A}_{i,r} \in \{0,1\}^{t\times t} | 1 \le r \le R\}$, and the local adjacency matrix $\mathbf{A}_{i,r}$ of node v_i in r-th relation is as follows

$$\mathbf{A}_{i,r}(p,q) = \begin{cases} 1, \ e_{pq} \in \mathcal{E}_r, \\ 0, \ \text{otherwise.} \end{cases}$$

Note that the order of nodes will not affect the embedding results much since IsoNN can alleviate the node order constraint posed by the adjacency matrix.

Structure Encoding with Multi-relational IsoNN Graph isomorphic layer in IsoNN adopts learnable kernel variable $\mathbf{K} \in \mathbb{R}^{m\times m}$ to learn the regional structure information. Here, to incorporate with multi-relational graph, we propose a multi-relational Graph Isomorphic Layer. we use relational kernels $\{\mathbf{K}_1, \ldots, \mathbf{K}_R\}$, where $\mathbf{K}_r \in \mathbb{R}^{m\times m}$ is for r^{th} relation. We can have multiple channels, so the learnable kernel for relation r is denoted as $\mathcal{K}_r \in \mathbb{R}^{c\times m\times m}$. Then features can be learned by

$$\mathcal{F}_{i,r}(l,j,s,h) = \|\mathbf{P}_j \mathcal{K}_r(l,:,:)\mathbf{P}_j^\top - \mathbf{A}_{i,r}(s:s+m, h:h+m)\|_F$$

where $\mathbf{P}_j \in \{0,1\}^{m\times m}$ is a permutation matrix from permutation set $\{\mathbf{P}_1, \cdots, \mathbf{P}_{m!}\}$. $\mathcal{F}_{i,r} \in \mathbb{R}^{c\times m!\times (t-m+1)\times (t-m+1)}$ because $\mathbf{A}_{i,r}$ generates $(t-m+1)\times(t-m+1)$ sub-matrices in total. After computing all possible permutations, we need to find the features made by the optimal permutation. Therefore, a min-pooling layer is adopted on dimension caused by permutation matrices to find the minimal value computed by optimal permutation

$$\bar{\mathcal{F}}_{i,r}(l,s,h) = \text{min-pooling}\left(\mathcal{F}_{i,r}(l,:,s,h)\right),$$

where $\bar{\mathcal{F}}_{i,r} \in \mathbf{R}^{c\times(t-m+1)\times(t-m+1)}$. To reduce the number of parameters, we adopt an average pooling layer for $\bar{\mathcal{F}}_{i,r}$ among all channels.

$$\hat{\mathbf{F}}_{i,r}(s,h) = \frac{\sum_{l=1}^{c} \bar{\mathcal{F}}_{i,r}(l,s,h)}{c},$$

where $\hat{\mathbf{F}}_{i,r} \in \mathbb{R}^{(t-m+1)\times(t-m+1)}$. Since values of the learned features can vary within a large range, we normalize those learned features. Moreover, differentiating which region contributes more can build better structural features. We reshape $\{\hat{\mathbf{F}}_{i,1}, \ldots, \hat{\mathbf{F}}_{i,R}\}$ into vectors and concatenate them as $\hat{\mathbf{F}}_i \in \mathbb{R}^{R\times(t-m+1)^2}$, then normalize it by

$$\mathbf{F}_i(r,s) = 1 - \frac{\exp(\hat{\mathbf{F}}_i(r,s))}{\sum_j \exp(\hat{\mathbf{F}}_i(r,j))}.$$

Here, we use '1' for subtraction because the smaller values show better matching between templates and subgraphs. Therefore, the local structure embedding $\mathbf{h}_i^{(s)}$ is

$$\mathbf{h}_i^{(s)} = \mathrm{ReLU}\left(\mathrm{MLP}\left(\bigcup_{r=1}^{R} \mathbf{F}_i(r,:)\right)\right).$$

where $\mathbf{h}_i^{(s)} \in \mathbb{R}^{d_h}$. The time cost for learning isomorphic features is $\mathcal{O}(cm!m^3R(t-m+1)^2)$, where $(t-m+1)^2$ is the number of submatrices, m^3 corresponds to matrix multiplication time cost, $m!$ is introduced in enumerating permutation matrices. Thus, by choosing small kernel size (small m) and few nodes (small t), the training is feasible.

3.3 Label-guided Contrastive Loss enhanced Optimization

Label prediction. To predict node labels, we concatenate attribute embedding and structure embedding together as the final node embedding. Formally, the prediction for node v_i can be made by

$$\hat{y}_i = \mathrm{MLP}\left(\mathrm{ReLU}\left(\mathrm{MLP}(\mathbf{h}_i^{(f)} \cup \mathbf{h}_i^{(s)})\right)\right).$$

The loss function for label prediction is weighted cross entropy.

$$\mathcal{L}_{ce} = -\sum_{i\in\mathcal{B}} \left(\gamma y_i \log\left(\hat{y}_i\right) + (1-y_i)\log(\hat{y}_i)\right),$$

where γ is the imbalance ratio of fraud labels ($y_i = 1$) to benign labels ($y_i = 0$), $\mathcal{B}$ is the training batch.

Label-guided Contrastive Loss. To better assist the optimization, we introduce two label-guided contrastive losses to provide extra guidance on learning node embeddings. As aforementioned, we separate the attribute and the structure since they would affect each other, which might make fraudsters harder to be discovered. Therefore, we contrast the attribute embedding and structure embedding separately. We set that positive pairs

are those with the same labels and negative pairs are those with different labels based on the training set. Here, we define the attribute label-guided contrastive loss as:

$$\mathcal{L}_{ac} = -\log \frac{\exp\left(f\left(\mathbf{h}_i^{(f)}, \mathbf{h}_{i^+}^{(f)}\right)/\tau\right)}{\exp\left(f\left(\mathbf{h}_i^{(f)}, \mathbf{h}_{i^+}^{(f)}\right)/\tau\right) + \sum_{i^- \in \mathcal{C}} \exp\left(f\left(\mathbf{h}_i^{(f)}, \mathbf{h}_{i^-}^{(f)}\right)/\tau\right)},$$

where i^+ means node v_{i^+} has $y_{i^+} = y_i$, i^- means node v_{i^-} has $y_{i^-} \neq y_i$, $\mathcal{C}$ is set containing negative instances, and τ is temperature. To get the negative instance set, we randomly select b nodes with different labels from the training set. Similarly, we can get $\mathcal{L}_{sc}$ with structure embedding. We formulate the objective function of DC-GNN as:

$$\mathcal{L} = \mathcal{L}_{ce} + \lambda_1 * \mathcal{L}_{ac} + \lambda_2 * \mathcal{L}_{sc}$$

where λ_1 and λ_2 control the weight of $\mathcal{L}_{ac}$ and $\mathcal{L}_{sc}$, respectively. The training goal is to minimize $\mathcal{L}$ and the parameters are updated by backpropagation.

4 Experiments

4.1 Experimental Settings

To validate the effectiveness of the proposed model, we use two real-world datasets – Yelp [16] and Amazon (AMZ) [13]. Yelp includes hotel and restaurant reviews filtered (spam) and recommended (legitimate) on Yelp platform. Amazon includes product reviews under the Musical Instruments category. The number of edges belonging to each relation is shown in Table 1. More information can be found at [2].

Table 1: Dataset Statistics

	# Nodes (Frauds%)	Relations		
Yelp	45,954 (14.5%)	R-U-R (49,315)	R-T-R (573,616)	R-S-R (3,402,743)
AMZ	11,944 (9.5%)	U-P-U (175, 608)	U-S-U (3,566,479)	U-V-U (1,036,737)

Comparison Methods and Evaluation Metrics Our baselines include GNN models and GNN-based fraud detection models. RGCN [18], GraphSAGE [4] are two general GNN models. For GNN-based models, we add FdGars [24] which is based on GCN, GraphConsis [12] which aggregates selected neighbors by similarity, DCI [25] which decouples representation learning and classification to enhance the performance, CARE-GNN [2] that uses a reinforcement learning module for neighborhood selection, FRAUDRE [30] which considers four types of inconsistencies in fraud detection.

Fraud detection suffers severe class imbalance. Therefore, AUC and average precision (AP) are our evaluation metrics.

Experimental setups We partition the datasets into training, validation, and test data. We choose $p \in \{60\%, 80\%\}$ of a dataset to train the model, and use 20% of the remaining data as the validation set. The rest is test data. We set the embedding size $d_h = 64$

Table 2: Fraudulent nodes similarity between its neighborhood. (FeatSim denotes feature similarity and LabelSim denotes label similarity. 2-hop via F/B means the 2-hop neighborhood retrieved via fraudulent nodes/benign nodes.)

datasets	Relation	FeatSim (1-hop)	LabelSim (1-hop)	FeatSim (2-hop via F)	LabelSim (2-hop via F)	FeatSim (2-hop via B)	LabelSim (2-hop via B)
Yelp	R-U-R	0.9906	0.9089	0.9955	0.9817	0.9940	0.5306
	R-T-R	0.9880	0.1764	0.9882	0.1989	0.9882	0.1932
	R-S-R	0.9878	0.1857	0.9879	0.1914	0.9879	0.1910
	ALL	0.9878	0.1838	0.9878	0.1737	0.9876	0.0955
Amazon	U-P-U	0.7107	0.1673	0.7069	0.1687	0.6121	0.0751
	U-S-U	0.6866	0.0558	0.6287	0.0638	0.6254	0.0499
	U-V-U	0.6969	0.0532	0.6239	0.0388	0.6153	0.0308
	ALL	0.6866	0.0722	0.6169	0.0684	0.6331	0.0528

for all comparison methods and set the same random seed to make a fair comparison. Parameters for graph transformer include layer number $L = 2$, and the number of heads = 8. In the individual attribute encoding, we set $K = 2$, but $|\mathcal{N}_i^{(2)}|$ is not a fixed number for all nodes. Therefore, we select the same number of neighbors from $\mathcal{N}_i^{(2)}$ to make it easy to train. We set the number of neighborhoods to 201 for Amazon and 61 for Yelp. For structure encoding, we only use 1 multi-relational graph isomorphic layer for simplicity. We set $t = 5$, the kernel size $m = 2$ and channel number $c = 10$ for Amazon, and $t = 20$, $m = 3$ and $c = 10$ for Yelp. In contrastive loss, $\tau = 0.001$, $b = 15$, and $\lambda_1 = 0.001$ and $\lambda_2 = 0$. for Amazon, and $\tau = 0.001$, $b = 5$, $\lambda_1 = 0$ and $\lambda_2 = 0.001$ for Yelp. The learning rate is 0.001 and 4096 is batch size. Adam optimizer is utilized.

For the baselines' implementation, we follow the setting reported in their papers. We use the codes of DG-Fraud [3] for GraphSAGE, FdGars and GraphConsis, and codes published by the authors for RGCN, CARE-GNN, DCI, FRAUDRE. For DC-GNN, we pre-train the individual attribute encoding module to get the ranking of the neighbhorhood so that we can reconstruct the adjacency matrices before training DC-GNN. All experiments are run on 256GB Linux server.

4.2 Camouflage Evidence in 2-hop Neighborhood

We use the same equation in [2] to calculate similarity scores. Table 2 shows the feature and label similarity scores among 1-hop and 2-hop neighborhood. According to the similarity scores of 1-hop neighbors, we observe that label similarity is as low as 0.1838 for Yelp and 0.0722 for Amazon, which indicates that fraudsters connect many benign nodes in both datasets, showing that connection camouflage is severe in almost all relations (except R-U-R in Yelp). Additionally, under such a low label similarity score, the feature similarity score is as high as 0.9878 for Yelp and 0.6866 for Amazon, which shows feature camouflage exists since fraudsters connect benign nodes sharing similar features. To have a better understanding of how camouflage affects the GNNs in 2-hop neighborhood. We have two scenarios: if the intermediate node is fraudulent (Via F) or benign (Via B). If the intermediate node is fraudulent, then both feature camouflage and

[3] https://github.com/safe-graph/DGFraud-TF2

Table 3: Classification Results. (The best scores are bold, the second best is underlined.)

Dataset	Metric	Training Percentage	RGCN	Graph-SAGE	FdGars	Graph-Consis	DCI	CARE-GNN	FRAU-DRE	DC-GNN
Yelp	AUC	60%	61.18	54.31	48.22	85.55	63.41	78.86	<u>86.08</u>	**88.13**
		80%	61.66	51.79	46.82	<u>86.60</u>	63.45	78.34	86.25	**88.18**
	AP	60%	23.46	17.32	14.59	54.18	21.07	42.17	<u>55.85</u>	**58.82**
		80%	24.33	14.87	14.19	56.10	17.53	41.73	<u>57.00</u>	**58.94**
Amazon	AUC	60%	18.84	73.78	40.75	89.37	88.52	92.66	<u>93.15</u>	**94.87**
		80%	18.97	75.88	41.48	90.52	86.04	93.02	<u>94.23</u>	**95.43**
	AP	60%	37.86	24.52	7.94	81.04	53.63	82.69	<u>84.37</u>	**86.95**
		80%	41.51	28.39	9.30	81.80	38.08	83.80	<u>85.47</u>	**87.56**

connection camouflage are still severe as the scores are similar to those of 1-hop. If the intermediate node is benign, two scores heavily drop compared with scores of 1-hop. This shows the information transmitted through benign nodes brings much misleading information, which further degrades the performance of traditional GNN models.

4.3 Overall Performance

The overall performance is shown in Table 3. RGCN and GraphSAGE and FdGars, are general graph neural networks and perform badly on both datasets. The main reason is traditional GNNs cannot handle the noises introduced by camouflage, especially when severe camouflage that exists in both datasets as illustrated in subsection 4.2. Among them, FdGars gets the worst performance since it cannot deal with multi-relations and is unable to filter out any noise brought by camouflage. DCI performs better in AUC metric than traditional GNN since it decouples the end-to-end classification into representation learning and classification, and the self-supervised graph learning in DCI captures the relatively comprehensive information about fraudsters. Moreover, when the number of training data increases, DCI's performance decreases, too. It shows the decoupling strategy of DCI is unable to learn well when facing more instances with camouflage. GraphConsis, CARE-GNN, and FRAUDRE have relatively good performances among all methods. FRADURE has the best performance among them since it considers four types of inconsistencies caused by camouflage and propagates additional inconsistency features. However, the noise brought by dissimilar neighbors is unable to remove. Compared with FRAUDRE, CARE-GNN is worse on two datasets. Especially on Yelp, AUC and AP are much lower. CARE-GNN selects nodes at each layer, which shows that the selection can filter out some neighbors connected by camouflaged edges successfully. However, the aggregation is based on node features. Thus, for Amazon, whose features provide more information than structures, it works well. but for Yelp, whose features do not provide much information (see discussion in subsection 4.5), it cannot work well. With more training data, its performance on Yelp is even worse. GraphConsis has better performance than CARE-GNN, but worse than FRAUDRE, since it can select k-hop neighbors and filter out most of the neighbors who do not share the same class with the central node. More training data brings better performance. DC-GNN outperforms all methods. For Yelp, it has around 2%, 2% improvement over the second-best

scores in AUC and AP, respectively. For Amazon, DC-GNN has about 1.5% and 2% improvement than the second best scores in AUC and AP, respectively. It means such a decoupled learning process can minimize the mutual influence between feature camouflage and relation camouflage and learn useful local structure information as much as possible. Overall, the performance shows the effectiveness of DC-GNN.

4.4 Discussion on Reconstructed Matrices

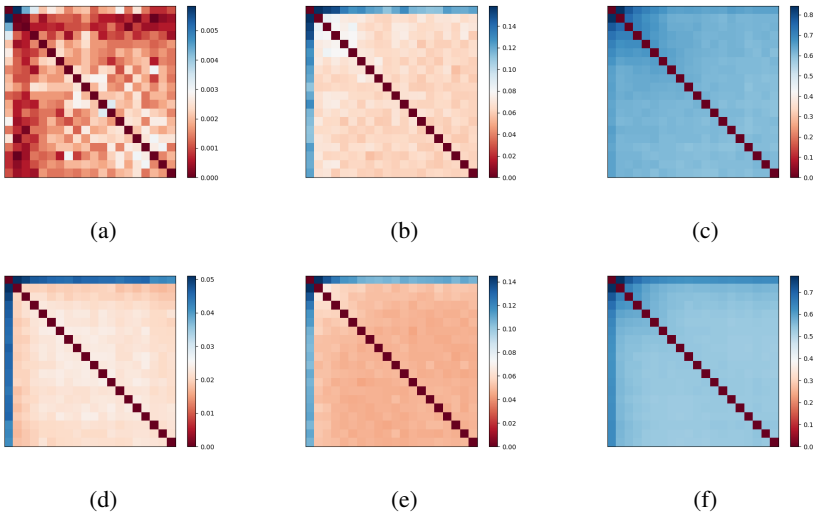
To better understand how the node selection works, we show the average reconstructed local adjacency matrices with 20 selected nodes on Yelp in Fig. 3. Specifically, Fig. 3a-3c show the reconstructed matrices for fraudulent nodes, while Fig. 3d-3f show reconstructed matrices for benign nodes. All reconstructed local adjacency matrices of fraudsters have clear different patterns from those for benign nodes. It illustrates the reconstructed matrices are able to give characteristics of fraudulent nodes. Furthermore, we show the distance and label similarity of selected nodes to central nodes in Table 4. Distance is bigger than 1, which shows selected nodes contain 2-hop neighbors. By selecting 2-hop neighbors, DC-GNN can reach more nodes and obtain richer information. The label similarity is low, which means the most of the selected neighbors are benign nodes. Thus, the reconstructed adjacency matrices are made of camouflaged edges, edges between benign nodes, and edges between fraudulent nodes. The camouflaged edges provide the links to benign nodes, while edges between benign nodes reveal the "community" that the fraudsters want to cheat. Therefore, the performance increases with the number of selected nodes increases. DC-GNN achieves the best performance with 20 nodes.

Table 4: Study on Selected Nodes of Yelp.

#Selected Nodes	AUC	AP	Distance	LabelSim
5	84.41	47.33	1.1311	0.2111
10	85.69	52.32	1.1955	0.2033
15	86.57	54.54	1.2197	0.2026
20	88.13	58.82	1.2306	0.2008

4.5 Ablation Study

To study the effectiveness of each module, we also conduct an ablation study with 60% training data. Fig. 4 shows the results. We first show how attribute and structure modules contribute seperately. Then, we add contrastive loss (CL) to attribute encoding and structure encoding separately and collectively. As illustrated in Fig. 4a, single attribute embedding or structure embedding cannot provide enough information to detect fraudsters in Amazon. By combining two modules, the performance gets a huge improvement. Interestingly, CL can better assist attributes than structure. It means attributes of fraudulent nodes are different from those of benign nodes. Meanwhile, structures of fraudulent nodes are extremely hard to distinguish from benign nodes, even getting worse performance with CL. As shown in Fig. 4b, combined embedding with CL is also the best performer on Yelp. However, different from Amazon, attribute embedding with CL gets much worse than it without CL, which means the attribute embedding in



(a) (b) (c)

(d) (e) (f)

Fig. 3: Reconstructed Local Adjcency Matrices of Yelp. (Fig. 3a- 3c for fraudulent nodes and Fig.3d- 3f for benign nodes.)

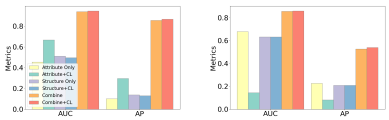
Yelp is hard to provide the right guidance under CL. Instead, the structure can provide useful guidance under CL to improve the model performance.

Number of Neighborhood. Here, we also discuss how the number of neighbors affects performance. As illustrated in Fig. 5, in both datasets, the difference is not much when choosing different numbers of neighbors for the individual attribute encoding module. Even for Amazon, DC-GNN has relatively high performance with 40 neighbors. Therefore, we can use a small-sized "context" to train the individual attribute encoding module, which can keep the training time within a reasonable range.

4.6 Parameter Analysis

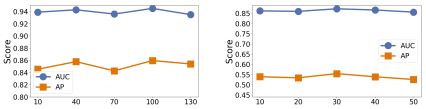
Parameters Analysis in Contrastive Loss. We have four parameters in contrastive loss, including the coefficient of attribute contrastive loss λ_1, coefficient of structure contrastive loss λ_2, temperature τ, and the number of negative instances b. The overall results are shown in Fig. 6.

- λ_1. We show the results that $\lambda_1 \in \{0.0001, 0.001, 0.01, 0.1\}$ in Fig. 6a and 6e. For Yelp, the performance has a sudden drop when $\lambda_1 = 0.01$. Scores of 0.0001 and 0.001 are similar and better than those of 0.1. For Amazon, the performance remains stable under different λ_1.
- λ_2. Similar to λ_1, we show the results that $\lambda_2 \in \{0.0001, 0.001, 0.01, 0.1\}$ in Fig. 6b and 6f. Figures show that the performance affected by structure information has a clear trend. When λ_2 decreases, the performance decreases as well. While for Yelp, the performance increases at $\lambda_2 = 0.001$ and then decreases, showing the best value for λ_2 is 0.001. For Amazon, the best value is 0.1.



(a) Amazon (b) Yelp

Fig. 4: Ablation Study



(a) Amazon (b) Yelp

Fig. 5: Number of Neighbhors

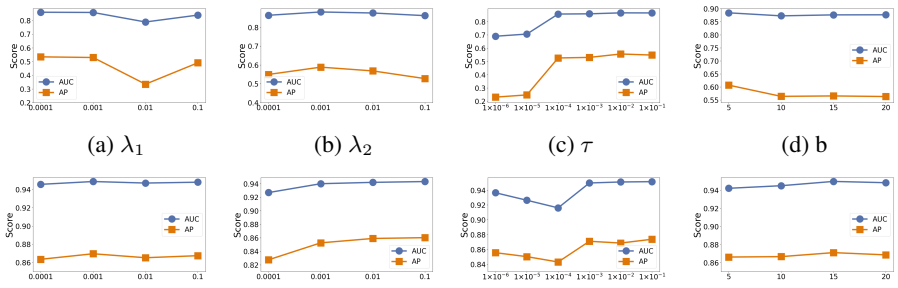
– τ. τ is an important parameter in the contrastive loss. we set $\tau \in \{1 \times 10^{-6}, 1 \times 10^{-5}, 1 \times 10^{-4}, 1 \times 10^{-3}, 1 \times 10^{-2}, 1 \times 10^{-1}\}$. From Fig. 6c, we observe a sharp increase when τ change from 1×10^{-5} to 1×10^{-4}, and have a slight increase afterwards. From Fig. 6g, the performance first goes down at 1×10^{-4} and then reach a peak at 1×10^{-3}. Therefore, a reasonable τ is curial for model performance.
– b. We choose the b from $\{5, 10, 15, 20\}$. As indicated in Fig. 6d and 6h. For Yelp, the best number is 5. When the number increases, the performance decreases first and remains steady afterward. For Amazon, when the number increases to 15, the performance reaches a peak.

5 Related Work

5.1 Graph Neural Networks and Graph Contrastive Learning

Graph Neural Networks (GNNs) [7, 4, 18] bring much easier computation along with better performance for graph-structured data. Generally, GNNs utilize the message-passing framework, which first aggregates all the message coming from the connected neighborhood, and then update the embedding for the central node. Prevailing methods to capture graph properties are in two granularities, including node [7, 18], subgraph [6, 27, 9]. From a node view, GCN [7], GraphSAGE [4] are some of the earliest works focusing on node classification. In heterogeneous graphs, many researchers use meta-path [3, 33] to construct homogeneous graphs, and then apply GNN layers or attention mechanisms on top of them. From the subgraph view, researchers claim subgraph can bring finer information on subgraph level [19, 6]. For instance, SubG-Con [6] better node embeddings by constructing node embeddings with subgraph embeddings. When using subgraph embeddings, a graph pooling layer like DGCNN [31], DiffPool [28] SAGPool [8], etc. are added after a GNN layer. These methods learn graph embedding with the help of node features. Different from them, IsoNN [14] can solely be based on the graph structure, but it cannot deal with large subgraphs.

Graph contrastive learning is often used in self-supervised learning, which requires positive samples and negative samples. Many works in literature contrast node embedding with its corresponding graph embedding or subgraph embedding [6, 22], some contrast subgraph with subgraph [29, 15]. However, in fraud detection, such contrast cannot be held due to camouflage. The neighborhood of fraudsters is noisy by having many benign nodes around. Therefore, previous contrast methods cannot be applied to fraud detection directly.



(a) λ_1 (b) λ_2 (c) τ (d) b

(e) λ_1 (f) λ_2 (g) τ (h) b

Fig. 6: Contrastive Loss Parameter Analysis Two datasets. Fig. 6a-6d shows the performance on Yelp, Fig. 6e- 6h performance on Amazon.

5.2 GNN-based Fraud Detection

Recent works have made much progress in applying GNN to fraud detection. FdGars [24] is the first paper using GCN [7] to detect fraudsters for online app review systems. Since only a very small portion of data has labels in fraud detection, SemiGNN [23] utilizes both labeled and unlabeled data in the loss function, while Player2Vec [32] construct multi-view networks from abundant information in the heterogeneous graph to enrich information. On the other hand, fraudsters often disguise themselves, therefore, some works like GraphConsis [12] and FRAUDRE [30] consider incorporating inconsistencies of node features and relations when applying GNN. Some other works [2, 11, 12] choose to filter out neighbors that do not share the same class before aggregation. CARE-GNN [2] utilizes reinforcement learning to distinguish nodes with camouflaged behaviors. Besides, some works still claim other issues, like class imbalance in GNN, incomplete information in the graph, etc. For example, PC-GNN [11] remedies the class imbalance problem by picking and choosing neighbors to aggregate at each layer. DCI [25] decouples the representation learning and classification to obtain performance gain. With many works in fraud detection, but rare papers are about the mutual effects of node features and graph structure under camouflage.

6 Conclusion

We study the fraud detection in graph setting with camouflage behaviors. Since feature camouflage and connection camouflage affect each other in traditional GNN learning. Therefore, we propose DC-GNN that decouples the learning process of traditional GNNs into attribute learning and structure learning, which adopts a graph transformer to learn attribute embedding based on four attributes and then select nodes from K-hop neighbors to reconstruct local adjacency matrices for all nodes. To learn structure embeddings, DC-GNN utilizes the multi-relational IsoNN. DC-GNN predicts labels by combining the attribute embedding and structure embedding. It also uses label-guided contrastive losses to enhance the performance. We conduct extensive experiments on two real-world datasets, and the results demonstrate the effectiveness of DC-GNN.

7 Acknowledgement

This work is partially supported by NSF through grants IIS-1763365 and IIS-2106972.

References

1. Tian Bian, Xi Xiao, Tingyang Xu, Peilin Zhao, Wenbing Huang, Yu Rong, and Junzhou Huang. Rumor detection on social media with bi-directional graph convolutional networks. In *AAAI*, 2020.
2. Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S Yu. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *CIKM*, pages 315–324, 2020.
3. Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of The Web Conference 2020*, pages 2331–2341, 2020.
4. William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, pages 1025–1035, 2017.
5. Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. Fraudar: Bounding graph fraud in the face of camouflage. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016.
6. Yizhu Jiao, Yun Xiong, Jiawei Zhang, Yao Zhang, Tianqi Zhang, and Yangyong Zhu. Sub-graph contrast for scalable self-supervised graph representation learning. In *ICDM*, pages 222–231. IEEE, 2020.
7. Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
8. Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *ICML*, 2019.
9. Xing Li, Wei Wei, Xiangnan Feng, Xue Liu, and Zhiming Zheng. Representation learning of graphs using graph convolutional multilayer networks based on motifs. *Neurocomputing*, 464:218–226, 2021.
10. Ting Liang, Guanxiong Zeng, Qiwei Zhong, Jianfeng Chi, Jinghua Feng, Xiang Ao, and Jiayu Tang. Credit risk and limits forecasting in e-commerce consumer lending service via multi-view-aware mixture-of-experts nets. In *WSDM*, pages 229–237, 2021.
11. Yang Liu, Xiang Ao, Zidi Qin, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. Pick and choose: A gnn-based imbalanced learning approach for fraud detection. In *Proceedings of the Web Conference 2021*, pages 3168–3177, 2021.
12. Zhiwei Liu, Yingtong Dou, Philip S Yu, Yutong Deng, and Hao Peng. Alleviating the inconsistency problem of applying graph neural network to fraud detection. In *SIGIR*, pages 1569–1572, 2020.
13. Julian John McAuley and Jure Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *WWW*, pages 897–908, 2013.
14. Lin Meng and Jiawei Zhang. Isonn: Isomorphic neural network for graph representation learning and classification. *arXiv preprint arXiv:1907.09495*, 2019.
15. Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1150–1160, 2020.
16. Shebuti Rayana and Leman Akoglu. Collective opinion spam detection: Bridging review networks and metadata. In *SIGKDD*, 2015.

17. Yuxiang Ren, Hao Zhu, Jiawei Zhang, Peng Dai, and Liefeng Bo. Ensemfdet: An ensemble approach to fraud detection based on bipartite graph. In *ICDE*, 2021.
18. Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.
19. Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Yuanxing Ning, Philip S Yu, and Lifang He. Sugar: Subgraph neural network with reinforcement pooling and self-supervised mutual information mechanism. In *WWW*, pages 2081–2091, 2021.
20. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
21. Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
22. Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.
23. Daixin Wang, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, Shuang Yang, and Yuan Qi. A semi-supervised graph attentive network for financial fraud detection. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 598–607. IEEE, 2019.
24. Jianyu Wang, Rui Wen, Chunming Wu, Yu Huang, and Jian Xion. Fdgars: Fraudster detection via graph convolutional networks in online app review system. In *WWW*, 2019.
25. Yanling Wang, Jing Zhang, Shasha Guo, Hongzhi Yin, Cuiping Li, and Hong Chen. Decoupling representation learning and classification for gnn-based anomaly detection. In *SIGIR*, pages 1239–1248, 2021.
26. Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2018.
27. Carl Yang, Mengxiong Liu, Vincent W Zheng, and Jiawei Han. Node, motif and subgraph: Leveraging network functional blocks through structural convolution. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 47–52. IEEE, 2018.
28. Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 4805–4815, 2018.
29. Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33:5812–5823, 2020.
30. Ge Zhang, Jia Wu, Jian Yang, Amin Beheshti, Shan Xue, Chuan Zhou, and Quan Z Sheng. Fraudre: fraud detection dual-resistant to graph inconsistency and imbalance. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 867–876. IEEE, 2021.
31. Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *AAAI*, 2018.
32. Yiming Zhang, Yujie Fan, Yanfang Ye, Liang Zhao, and Chuan Shi. Key player identification in underground forums over attributed heterogeneous information network embedding framework. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 549–558, 2019.
33. Jianan Zhao, Xiao Wang, Chuan Shi, Binbin Hu, Guojie Song, and Yanfang Ye. Heterogeneous graph structure learning for graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4697–4705, 2021.