H^2-FDetector: A GNN-based Fraud Detector with Homophilic and Heterophilic Connections

Fengzhao Shi
shifengzhao@iie.ac.cn
School of Cyber Security
University of Chinese Academy of Sciences
China
Institute of Information Engineering
Chinese Academy of Sciences
China

Yanan Cao*
caoyanan@iie.ac.cn
Institute of Information Engineering
Chinese Academy of Sciences
China
School of Cyber Security
University of Chinese Academy of Sciences
China

Yanmin Shang†
shangyanmin@iie.ac.cn
Institute of Information Engineering
Chinese Academy of Sciences
China
School of Cyber Security
University of Chinese Academy of Sciences
China

Yuchen Zhou
zhouyuchen@iie.ac.cn
School of Cyber Security
University of Chinese Academy of Sciences
China
Institute of Information Engineering
Chinese Academy of Sciences
China

Chuan Zhou
zhouchuan@amss.ac.cn
Academy of Mathematics and Systems Science
Chinese Academy of Sciences
China
School of Cyber Security
University of Chinese Academy of Sciences
China

Jia Wu
jia.wu@mq.edu.au
School of Computing
Macquarie University
Sydney, NSW 2113, Australia

ABSTRACT

In the fraud graph, fraudsters often interact with a large number of benign entities to hide themselves. So, there are not only the homophilic connections formed by the same label nodes (similar nodes), but also the heterophilic connections formed by the different label nodes (dissimilar nodes). However, the existing GNN-based fraud detection methods just enhance the homophily in fraud graph and use the low-pass filter to retain the commonality of node features among the neighbors, which inevitably ignore the difference among neighbor of heterophilic connections. To address this problem, we propose a Graph Neural Network-based Fraud Detector with Homophilic and Heterophilic Interactions (H^2-FDetector for short). Firstly, we identify the homophilic and heterophilic connections with the supervision of labeled nodes. Next, we design a new information aggregation strategy to make the homophilic connections propagate similar information and the heterophilic connections propagate difference information. Finally, a prototype prior is introduced to guide the identification of fraudsters. Extensive experiments on two real public benchmark fraud detection tasks demonstrate that our method apparently outperforms state-of-the-art baselines.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; *Machine learning*; • **Security and privacy** → *Social network security and privacy*.

KEYWORDS

Fraud Detection, Graph Neural Networks, Homophily, Heterophily

ACM Reference Format:
Fengzhao Shi, Yanan Cao, Yanmin Shang, Yuchen Zhou, Chuan Zhou, and Jia Wu. 2022. H^2-FDetector: A GNN-based Fraud Detector with Homophilic and Heterophilic Connections. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3485447.3512195

1 INTRODUCTION

As a basic network service, fraud detection is widely used in network security [7], e-commerce [3, 10, 15, 36], review management [5, 12, 22] and other critical areas. Recently, graph-based fraud detection approaches [19, 31], specifically GNN-based ones, have escalated lots of attention in both academic and industrial communities. These methods utilize relations among entities to reveal the suspiciousness of these entities at the fraud graph level, based on the assumption that fraudsters with the same goal tend to connect with each other.

However, many investigations [6, 9, 11, 20] discover that fraudsters often camouflage themselves via connecting with many benign entities to alleviate the suspiciousness. Take spammers for example, they would employ benign accounts to post their spam

*Corresponding Author
†Corresponding Author

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04.
https://doi.org/10.1145/3485447.3512195

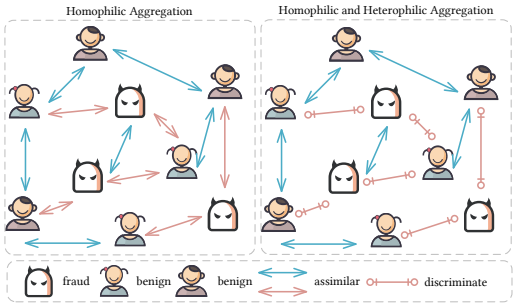


Figure 1: Homophily vs. Homophily and Heterophily.

reviews so that there will be many connections between the spam review and the benign users. So, there are actually two kinds of entity connections in a fraud graph: one is among nodes with the same labels, called the homophilic connections; the other is among nodes with different labels (fraudster and benign entities), called the heterophilic connections. As shown in Figure 1 (a), most of the existing GNN-based fraud detection methods [6, 17, 20] only utilize the homophily in fraud detection and ignore the difference among neighbors with different labels. In the GNN aggregating procedure, they use the low-pass to retain the commonality of node features among the neighbors and assimilate the representations of connected nodes even if they have different labels. This information aggregating strategy would introduce noises into node representation learning and weaken the performance of fraud detection.

To address this issue, our paper aims to simultaneously model the homophilic and heterophilic connections (**H^2-connection** for short) in the fraud graph, assimilating the nodes with homophilic connections and discriminating the nodes with heterophilic connections (Figure 1 (b)). In order to achieve this goal, we should solve the following challenges: **1) How to identify the H^2-connection in a fraud graph?** In the training dataset, the labels of the nodes are given, so we can easily get the supervision signal of whether a connection is homophilic or heterophilic. This connection classifier could be modeled jointly with the fraud detector. **2) How to design the GNN aggregation strategy under the mixture of H^2-connection?** During the process of GNN aggregation, the homophilic connection should propagate similar information, while the heterophilic connection should propagate dissimilar information. So, the key point is how to model and utilize the similar information and dissimilar information among nodes. **3) How to utilize the category features of all known fraudsters to identify new fraudsters?** In practical application, some fraudsters may be trapped in overmany benign entity neighborhoods, and these fraudsters can only obtain dissimilarity information from benign neighbors but lack the information of similarity from other fraudsters. So, we aim to capture the category features to help reveal the suspicious of the central node.

Motivated by above intuitions, we proposes a Graph Neural Net-work-based Fraud Detector with Homophilic and Heterophilic Interactions (H^2-FDetector for short). Firstly, we design a neural network to identify H^2-connection with the supervision of labeled nodes in the training set. Then, we design the aggregation strategy based on GAT. For the homophilic connections, we use the typical way of GAT to conduct the self-attention. For the heterophilic ones, we use the opposite representation of neighbors to conduct the self-attention. In this way, the learned representation of nodes with the same labels tend to be more similar while those with different labels tend to be dissimilar. Finally, a category prototype is taken as the category characteristic of fraudsters in GNN aggregation to provide a prior guidance.

We highlight the advantages of H^2-FDetector as follows:

- As far as we known, we are the first to simultaneously consider the influence of homophilic and heterophilic connections for graph-based fraud detection.
- We design a novel graph neural network, which can not only aggregate similarity information for the homophilic connection, but also dissimilar information for the heterophilic connection.
- Experiments conducted on two real public benchmark datasets verify the effectiveness of the proposed framework. In particular, on the most important recall metric of fraud detection, the performance has been remarkably improved by our method.

2 PROBLEM FORMULATION

In this section, we first give the conception of homophily and heterophily. Then, we define the H^2-connection multi-relation graph and the graph-based fraud detection problem.

2.1 Preliminaries

Definition 1. Homophily and heterophily. For a graph, if the two nodes connected by an edge belong to the same category label, then the edge (connection) is homophilic. Otherwise, this edge is heterophilic. The graph composed of the homophilic edges is called the homophilic graph, and the graph composed of the heterophilic edges is called the heterophilic graph. In particular, fraud graph has homophilic and heterophilic edges at the same time.

2.2 Problem Statement

Definition 2. H^2-connection multi-relation graph. Given a graph $\mathcal{G} = \{\mathcal{V}, X, \{\mathcal{E}_r^+, \mathcal{E}_r^-\}|_{r=1}^R, Y\}$, $\mathcal{V} = \{v_1, v_2, ..., v_N\}$ is the set of nodes, N is the number of nodes; $X = \{x_1, x_2, ..., x_N\}$ is the set of node features, $x_i \in \mathbb{R}^d$ is i^{th} node feature, d is the dimension of feature; $\{\mathcal{E}_r^+, \mathcal{E}_r^-\}$ is the set of edges with a relation $r \in \{1, ..., R\}$, $\mathcal{E}_r^+$ represents homophilic edges set, $\mathcal{E}_r^-$ represents heterophilic edges set, $\mathcal{E}_r^+ \cap \mathcal{E}_r^- = \emptyset$ and Y is the set of labels corresponding to the nodes. $\mathcal{G}$ is directed.

Definition 3. Graph-based fraud detection. The graph-based fraud detection problem is defined on H^2-connection multi-relation graph $\mathcal{G} = \{\mathcal{V}, X, \{\mathcal{E}_r^+, \mathcal{E}_r^-\}|_{r=1}^R, Y\}$, which has been formulated in definition 2. The graph-based fraud detection problem is a semi-supervised binary node classification problem on the graph. Graph-based fraud detectors are trained based on the labeled node information along with the graph composed of homophilic and heterophilic edges under multi-relations. The trained models are then used to predict the suspiciousness of unlabeled nodes. It is worth noting that the homophily and heterophily of edges are learned through model.

3 METHODOLOGY

In this section, we present H²-FDetector framework. Firstly, we give an overview of the whole framework. Then we detail three components of model in Section 3.2, 3.3 and 3.4 respectively. Finally, we conclude the model loss and training process.

3.1 Overview

H²-FDetector includes multi-layer convolution and each layer is made up of three components: H²-connection identification, H²-connection aggregation and prototype extraction. The pipeline of H²-FDetector is shown in Figure 2. For a central node, the connections with neighbor nodes are judged as homophilic or heterophilic by H²-connection identification. Then, we devise an H²-connection aggregation to aggregate neighbor embeddings according to homophily or heterophily of each neighbor connection under each relation. Finally, we introduce a prior information for detecting fraudster with global prototype.

3.2 H²-connection Identification

The labels of many nodes in the fraud graph are unknown, so we can't directly judge whether these edges are homophilic or heterophilic according to whether the labels are same or not. Based on the assumption that nodes with the same label are similar and nodes with different labels are dissimilar, here,we design a H²-connection identification module with an end-to-end pattern to measure the similarity or difference between nodes, which can avoid the introduction of threshold hyperparameter. Details are as follows:

Formally, given an H²-connection multi-relation graph $\mathcal{G} = \{\mathcal{V}, X, \{\mathcal{E}_r^+, \mathcal{E}_r^-\}|_{r=1}^R, Y\}$, we abbreviate it as $\mathcal{G} = \{\mathcal{V}, X, \{\mathcal{E}_r\}|_{r=1}^R, Y\}$ due to lacking connection types, our goal is to obtain the homophily or heterophily of $\mathcal{E}_r$, where $\mathcal{E}_r = \mathcal{E}_r^+ \cup \mathcal{E}_r^-$ is the edge set of r^{th} relation and $\mathcal{E} = \bigcup_{r=1}^R \mathcal{E}_r$ is the set of all edges. $H^{\{l-1\}} = \{h_1^{\{l-1\}}, h_2^{\{l-1\}}, ..., h_N^{\{l-1\}}\}$ is the set of node embedding at layer $l-1$, $h_i^{\{l-1\}} \in \mathbb{R}^{d_{l-1}}$ is i^{th} node embedding and d_{l-1} is the dimension, $H^{\{0\}} = X$. For each edge $e_{uv} \in \mathcal{E}$, its head node and tail node are u and v respectively.

For each convolution layer l, the input of the H²-connection identification module comes from transformation of the upper layer, for example, for an edge $e_{uv} \in \mathcal{E}$, its input is $(\bar{h}_u^{\{l\}}, \bar{h}_v^{\{l\}})$:

$$\bar{h}_u^{\{l\}} = \sigma(W_t^{\{l\}} h_u^{\{l-1\}}) \tag{1}$$

$$\bar{h}_v^{\{l\}} = \sigma(W_t^{\{l\}} h_v^{\{l-1\}}) \tag{2}$$

where $h_u^{\{l-1\}}$ and $h_v^{\{l-1\}}$ are the embedding of u and v at layer $l-1$, $W_t^{\{l\}} \in \mathbb{R}^{d_l \times d_{l-1}}$ is the parameter matrix, $\sigma(.)$ is nonlinear activation function.

For getting more comprehensive perception information, we take the concatenation and difference between transformed embeddings $\bar{h}_u^{\{l\}}$ and $\bar{h}_v^{\{l\}}$ as the input of a classifier, here we use a one-layer Multi-layer Perceptron (MLP) as basic classifier, with *tanh* activation:

$$m_{uv}^{\{l\}} = tanh(W_c^{\{l\}}[\bar{h}_u^{\{l\}} || \bar{h}_v^{\{l\}} || (\bar{h}_u^{\{l\}} - \bar{h}_v^{\{l\}})]) \tag{3}$$

where $W_c^{\{l\}} \in \mathbb{R}^{3d_l}$ is the parameter matrix of classifier, [.||.] is concatenate operation. Then, We get the connection types by taking sign on $m_{uv}^{\{l\}}$:

$$c_{uv}^{\{l\}} = SIGN(m_{uv}^{\{l\}}) \tag{4}$$

where if $c_{uv}^{\{l\}}$ is 1, the edge $e_{uv} \in \mathcal{E}$ is homophilic, if $c_{uv}^{\{l\}}$ is -1, the edge $e_{uv} \in \mathcal{E}$ is heterophilic.

Finally, applying the above process to each edge $e_{uv} \in \mathcal{E}$, we can get the types of all edges in graph $\mathcal{G}$:

$$C^{\{l\}} = \{c_{uv}^{\{l\}}\}_{e_{uv} \in \mathcal{E}} \tag{5}$$

In the above process, $m_{uv}^{\{l\}}$ is the core of the whole module, and we add an auxiliary loss [35] to learn it by introducing the supervision signal from the known label nodes.

$$\mathcal{L}_{HI}^{\{l\}} = \frac{1}{\mathcal{E}_t} \sum_{e_{uv}}^{\mathcal{E}_t} max(0, 1 - y_{uv}^{\{e\}} m_{uv}^{\{l\}}) \tag{6}$$

where $\mathcal{E}_t$ is the edge set whose head nodes and tail nodes has been labeled, $y_{uv}^{\{e\}}$ represents the types of the corresponding edge. For each edge $e_{uv} \in \mathcal{E}$, if the labels of u and v are same, $y_{uv}^{\{e\}} = 1(\, e_{uv} \in \mathcal{E}$ is homophilic), otherwise, $y_{uv}^{\{e\}} = -1(\, e_{uv} \in \mathcal{E}$ is heterophilic).

3.3 H²-connection Aggregation

After acquiring connection types, H²-FDetector needs to aggregate neighbor nodes from both homophilic and heterophilic neighbors. According to [1], the sum of neighborhood representations makes the representations become similar, which is suitable for homophilic connections. While the difference of between node features and neighborhood features makes the representations become discriminative, which suits heterophilic connections. We use it as a basic aggregation strategy. However, it is hard to devise appropriate influence weight parameters for different neighbor nodes under homophily and heterophily. This is because in the case of the homophilic edges, the influence parameters represent the weight of similar influence; In the case of heterophilic edges, the influence parameters represent the weight of different influence. The existing self-attention mechanism[18, 29, 38] is essentially a sum of neighbor information and not suitable for heterophilic neighbor nodes. To tackle this issue, we design an H²-connection aggregation strategy with a new self-attention mechanism which can adapt to homophilic and heterophilic neighbors simultaneously.

Specifically, given r^{th} relation subgraph $\mathcal{G}_r = \{\mathcal{V}, X, \{\mathcal{E}_r\}, Y\}$, $\mathcal{E}_r = \mathcal{E}_r^+ \cup \mathcal{E}_r^-$ is the edge set of graph $\mathcal{G}_r$. We define $\mathcal{N}_r(v)$ as the neighbor set of node v in graph $\mathcal{G}_r$. Given a central node v and its arbitrary neighbor $u \in \mathcal{N}_r(v)$, we first transform their embedding using a linear transformation, parameterized by weight matrix $W_r^{\{l\}} \in \mathbb{R}^{d_l \times d_{l-1}}$. Then, we import the H²-connection aggregation strategy and compute the importance score of u to v with self-attention by considering the connection type $c_{uv}^{\{l\}}$:

$$e_{uv}^{\{l\},r} = a^{\{l\},r}[W_r^{\{l\}} h_v^{\{l-1\}} || c_{uv}^{\{l\}} W_r^{\{l\}} h_u^{\{l-1\}}] \tag{7}$$

where $a^{\{l\},r} \in \mathbb{R}^{1 \times 2d_l}$ is a wight vector.

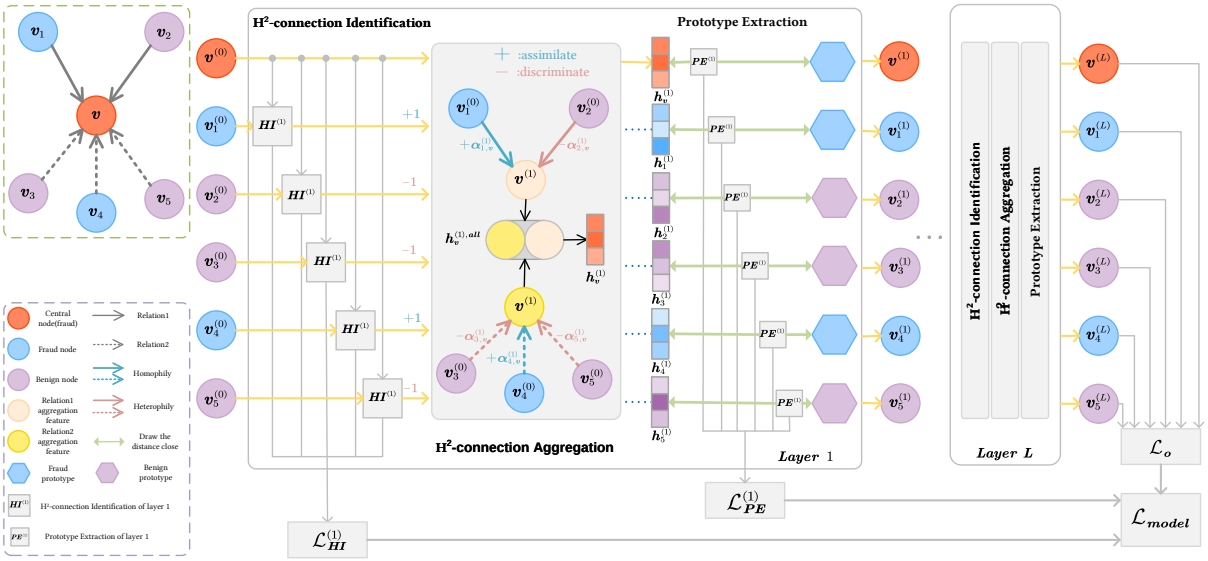


Figure 2: The Aggregation Process of Proposed H^2-FDetector at the Training Phase.

Then we compute attention coefficient between node v and its neighbor u by normalizing importance coefficient of all neighbors:

$$\alpha_{u,v}^{\{l\},r} = \frac{exp\{LeakyReLU(e_{uv}^{\{l\},r})\}}{\sum_{k\in\mathcal{N}_r(v)} exp\{LeakyReLU(e_{kv}^{\{l\},r})\}} \tag{8}$$

Next, we aggregate different neighbors with attention and connection type $c_{uv}^{\{l\}}$ to obtain the embedding $h_v^{\{l\},r}$ of central node v under r^{th} relation subgraph:

$$h_v^{\{l\},r} = \sigma\left(\sum_{u\in\mathcal{N}_r(v)} \alpha_{u,v}^{\{l\},r} c_{uv}^{\{l\}} W_r^{\{l\}} h_u^{\{l-1\}}\right) \tag{9}$$

For getting more information, we extend self-attention mechanism to multi-head form, similarly to [25, 33]. Specifically, K independent attention mechanisms execute the transformation of Eq. 9, and then their features are concatenated, resulting in the following output embedding:

$$h_v^{\{l\},r} = \Big\|_{k=1}^{K} \sigma\left(\sum_{u\in\mathcal{N}_r(v)} \alpha_{u,v}^{\{l\},r,k} c_{uv}^{\{l\}} W_r^{\{l\},k} h_u^{\{l-1\}}\right) \tag{10}$$

where $W_r^{\{l\},k} \in \mathbb{R}^{\frac{d_l}{k}\times d_{l-1}}$.

It is noteworthy that on the final layer of the network, we perform multi-head attention with averaging instead of concatenation to keep model sensible:

$$h_v^{\{L\},r} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{u\in\mathcal{N}_r(v)} \alpha_{u,v}^{\{L\},r,k} c_{uv}^{\{L\}} W_r^{\{L\},k} h_u^{\{L-1\}}\right) \tag{11}$$

Finally, we concatenate the R embeddings aggregated from corresponding relation subgraph and transfer it to a low-dimension embedding, and obtain the embedding $h_v^{\{l\}}$ of central node v :

$$h_v^{\{l\},all} = \Big\|_{r=1}^{R} h_v^{\{l\},r} \tag{12}$$

$$h_v^{\{l\}} = W_d^{\{l\}} h_v^{\{l\},all} \tag{13}$$

where $W_d^{\{l\}} \in \mathbb{R}^{d_l\times Rd_l}$ is weight matrix.

3.4 Prototype Extraction

Based on H^2-connection aggregation, we can obtain the representation of node. However, some fraudsters are trapped in overmany benign entity neighborhoods, and these fraudsters can only obtain the information of inter-class dissimilarity from benign neighbors but lack the information of intra-class similarity from other fraudsters. To solve the problem, we introduce a category information with prototype. We adapt prototype for two reasons: first, the prototype is in line with our original design intention which uses labeled data to find the approximate category center of each class, and then makes samples within the class more similar by shortening the distance between each sample and the prototype; second, the prototype can be easily plugged into our entire model architecture.

Specifically, we first get the prototypes of fraud and benign classes respectively based on training set:

$$prototype_{fraud}^{\{l\}} = \frac{1}{|\mathcal{V}_f|}\sum_{v\in\mathcal{V}_f} h_v^{\{l\}} \tag{14}$$

$$prototype_{benign}^{\{l\}} = \frac{1}{|\mathcal{V}_b|}\sum_{v\in\mathcal{V}_b} h_v^{\{l\}} \tag{15}$$

where $\mathcal{V}_f$ and $\mathcal{V}_b$ are the set of fraud nodes and set of benign nodes respectively in the training set.

Then, for each node $v \in \mathcal{V}_t$, we measure the distances between it and the two prototypes respectively. The distance function is the

Euclidean distance that is widely-used in latent space:

$$\mathcal{D}_f^{\{l\}}(v) = \left\| h_v^{\{l\}} - prototype_{fraud}^{\{l\}} \right\|_2 \tag{16}$$

$$\mathcal{D}_b^{\{l\}}(v) = \left\| h_v^{\{l\}} - prototype_{benign}^{\{l\}} \right\|_2 \tag{17}$$

If the node is a fraudster, it should be close to the fraud prototype and far from the benign entity prototype. On the contrary, if the node is a benign entity, it should be close to the benign entity prototype and far away from the fraudster prototype. Based on this assumption, we use cross entropy loss to shorten the distance between node v and its corresponding label prototype, and to lengthen the distance between node v and its opposite label prototype.

$$\mathcal{L}_{PE}^{\{l\}} = -\sum_{v \in \mathcal{V}_t} \left[y_v log(q_v^{\{l\}}) + (1 - y_v) log(1 - q_v^{\{l\}}) \right] \tag{18}$$

$$q_v^{\{l\}} = softmax\left(-\mathcal{D}_{C(v)}^{\{l\}}(v) \right) \tag{19}$$

where $C(v)$ is the label of v.

3.5 Training

After stacking multiple layers, we take the output of the last layer as the final embeddings of nodes. Based on this, we regard the fraud detection problem as a node classification problem on the graph, and use the cross entropy loss to model it.

$$\mathcal{L}_o = -\sum_{v \in \mathcal{V}_t} [y_v log(p_v) + (1 - y_v) log(1 - p_v)] \tag{20}$$

$$p_v = softmax(h_v^{\{L\}}) \tag{21}$$

where $h_v^{\{L\}}$ is the embedding of node v on last layer.

To sum up, the overall loss function of our model H^2-Detector is formulated as Eq. 22, where γ_1 and γ_2 are the balance parameters.

$$\mathcal{L} = \mathcal{L}_o + \gamma_1 \sum_{l-1}^{L} \mathcal{L}_{HI}^{\{l\}} + \gamma_2 \sum_{l=1}^{L} \mathcal{L}_{PE}^{\{l\}} \tag{22}$$

It is worth noting that to reduce the influence of sample imbalance (the sample of benign entities is significantly more than that of fraudsters), we employ under-sampling technique to train H²-FDetector. Specifically, we randomly sample the same number of majority class instances as the minority class. Then we use the sampled instances to compute loss and optimize H²-FDetector. We call this training method as random sample training.

The overall training algorithm is summarized in Algorithm 1. Given an H²-connection multi-relation graph $\mathcal{G}$. We first get the connection types of all edges according to Eq. 3 (Line 4). Then we aggregate neighbor information for each node according to connection type (from Line 7 to Line 11). Next, we calculate fraud and benign prototypes (Line 12 and Line 13) and enhance the global label prior information of each node (Line 14 and Line 15). In addition, to reduce the influence of sample imbalance, we compute the loss with random sample training (Line 5, Line 14 and Line 17).

Algorithm 1: The training process of H²-FDetector

Input: an H²-connection multi-relation graph without connection type label $\mathcal{G} = \{\mathcal{V}, X, \{\mathcal{E}_r\}|_{r=1}^{R}, Y\}$. A connection type label for training $Y_t^{\{e\}}$. Training epochs N_{epoch}. Number of layers L.

Output: The vector representation for each node in $\mathcal{V}$.

1 Initialize $H^0 \leftarrow X$;
2 **for** $e = 1, ..., N_{epoch}$ **do**
3 **for** $l = 1, ..., L$ **do**
4 $C^{\{l\}} \leftarrow$ construct connection types (section 3.2);
5 $C_{train}^{\{l\}}, Y_{train}^{\{e\}} \leftarrow$ balace sample from $C^{\{l\}}$ and $Y_t^{\{e\}}$;
6 $\mathcal{L}_{HI}^{\{l\}} \leftarrow$ Eq. 6 $\forall m_{uv}^{\{l\}}$ *corresponding* $c_{uv}^{\{l\}} \in C_{train}^{\{l\}}$, $y_{uv}^{\{e\}} \in Y_{train}^{\{e\}}$;
7 **for** $v \in \mathcal{V}$ **do**
8 **for** $r = 1, ..., R$ **do** $h_v^{\{l\},r} \leftarrow$ Eq. 10;
9 $h_v^{\{l\},all} \leftarrow$ Eq. 12;
10 $h_v^{\{l\}} \leftarrow$ Eq. 13;
11 **end**
12 $prototype_{fraud}^{\{l\}} \leftarrow$ Eq. 14;
13 $prototype_{benign}^{\{l\}} \leftarrow$ Eq. 15;
14 $H_{PE}^{\{l\}}, Y_{PE}$ balace sample from $H^{\{l\}}, Y$;
15 $\mathcal{L}_{PE}^{\{l\}} \leftarrow$ Eq. 18 $\forall h_v^{\{l\}} \in H_{PE}^{\{l\}}, y_v \in Y_{PE}, prototype_{fraud}^{\{l\}}, prototype_{benign}^{\{l\}}$;
16 **end**
17 $H_{train}^{\{L\}}, Y_{train} \leftarrow$ balace sample from $H^{\{L\}}, Y$;
18 $\mathcal{L}_o \leftarrow$ Eq. 20 $\forall h_v^{\{L\}} \in H_{train}^{\{L\}}, y_v \in Y_{train}$;
19 Back-propagation to update parameters;
20 **end**
21 **return** $H^{\{L\}}$

4 EXPERIMENTS

In this section, we would answer the following research questions:

- **RQ1**: Is heterophily prevalent in fraud detection?
- **RQ2**: Does H²-FDetector outperform the state-of-the-art methods for graph-based fraud detection?
- **RQ3**: How do the modules of heterophily perception and prototype extraction benefit the prediction?
- **RQ4**: What is the performance of H²-FDetector with respect to different hyperparameter?
- **RQ5**: Can H²-FDetector further effectively shorten the distance between fraudsters while extend the distance between fraudsters and benign entities?

4.1 Experiment Setup

4.1.1 Datasets. We conduct experiments on two real-world fraud detection datasets to evaluate the effectiveness of H²-FDetector: YelpChi dataset [24] and Amazon dataset [21]. The **YelpChi** dataset includes hotel and restaurant reviews filtered (spam) and recommended (legitimate) by Yelp, which can conduct spam review detection task. The nodes of YelpChi dataset are reviews with 32 handcrafted features and the dataset includes three relations: 1)

Table 1: The Statistic of Datasets.

dataset	#nodes (**fraud%**)	relation	#relations	class	#class
YelpChi	45,954 (**14.53%**)	R-U-R	98,630	Positive	6,677
		R-T-R	1,147,232	Negative	39,277
		R-S-R	6,805,486		
Amazon	11,944 (**6.87%**)	U-P-U	351,216	Positive	821
		U-S-U	7,132,958	Negative	7,818
		U-V-U	2,073,474	Unlabeled	3,305

R-U-R that connects reviews posted by the same user, 2) R-S-R that connects reviews under the same product with the same star rating, 3) R-T-R that connects two reviews under the same product posted in the same month. The **Amazon** dataset includes product reviews under the Musical Instruments category. The nodes of Amazon dataset are users with 25 handcrafted features and the dataset includes three relations: 1) U-P-U that connects users reviewing at least one same product, 2) U-S-U that connects users having at least one same star rating within one week, 3) U-V-U that connects users with top 5% mutual review text similarities (measured by TF-IDF) among all users. The statistic of datasets is shown in table 1.

4.1.2 Baselines. We compare with several state-of-the-art GNN-based methods to verify the effectiveness of H^2-FDetector in fraud detection.

Traditional GNNs: GCN [13], **SGC** [34] and **GAT** [28] treat all connections as homophilic and do not consider heterophilic connections.

Improved GNNs: GPRGNN [4] and **FAGCN** [1] explore homophilic and heterophilic connections in graph by designing the new GNNs aggregation mechanism based on generalized pagerank and self-gating mechanism respectively. We use the traditional GNNs or improved GNNs to get the representation of nodes in fraud graph, and then use the classifiers to identify fraudsters.

GNNs-based Fraud Detection: CARE-GNN [6] and **PC-GNN** [17] are the state-of-the-art GNN-based methods for fraud detection. They discover that fraudsters often disguise themselves by connecting with benign entities, so they use the under-sampling methods to reduce this noise and use traditional GNNs aggregate neighbors by considering all the connections are homophilic.

H^2-FDetector: our proposed method. We also derive two variants of H^2-FDetector to comprehensively compare and analyze the performances of its each component. They are,

H^2-FDetector$_{\backslash HP}$: removes the heterophily perception process and set $c_{uv}^{\{l\}} = 1, (u, v) \in \mathcal{E}, l \in \{1, ..., L\}$).

H^2-FDetector$_{\backslash PE}$: removes the prototype extraction process.

4.1.3 Experimental Setting. For H^2-FDetector, the learning rate is set to 0.1 and weight decay is 0.00005, the dimension of node embedding is set to 8 and the number of head is 4, the number of layers is set to 2 and N_{epoch} is 1000. To avoid overfitting, we use dropout mechanism for all methods and the dropout rate of H^2-FDetector is set to 0.1. For YelpChi dataset, the two hyperparameters γ_1 and γ_2 are both set to 1.2, and for Amazon dataset, they are set to 0.4 and 1.4 respectively. The GCN, SGC, GAT, GPRGNN and FAGCN are tuned the best parameters for different dataset with grid search. For CARE-GNN, we use the best parameters introduced by authors. For PC-GNN, we tune the parameters on YelpChi Dataset due to the difference of two versions of the dataset and use the parameters introduced by authors on Amazon Dataset. The division of datasets are similar as [17]. In addition, the original GCN, SGC, GAT, GPRGNN and FAGCN suffer from class imbalance problem, and to tackle this problem, we use the random sample training for these methods. All methods are optimized with Adam optimizer.

4.1.4 Implementation. For GCN, SGC and GAT, we implement them based on DGL [32]. For GPRGNN, FAGCN, CARE-GNN and PC-GNN, we carry out the source code provided by authors. We implement H^2-FDetector in Pytorch [23]. All models are running on Python3.6.12, 1 NVIDIA Tesla V100 GPU, 629GB RAM, 2.20GHz Intel Xeon E5-2650 CPU.

4.1.5 Evaluation Metric. The fraud detection datasets are inherently unbalanced, and although fraudsters (positive instances) are in the minority, they are more concerned. In this paper, we use four evaluation metrics to evaluate all models: **Recall**, **AUC-ROC (AUC)**, **F1-macro** and **GMean**. Among them, **Recall** is the most important for fraud detection.

4.2 Heterophily Evidence (RQ1)

To answer the RQ1, we calculate the heterophily ratio of heterophilic edges to all adjacent edges of each fraud node in different fraud graphs, and count the proportion of the number of fruad nodes with the corresponding heterophily ratio to all the fruad nodes in the whole graph from low to high (Figure 3). We observe that these relation subgraphs, except R-U-R relation of YelpChi, include a large number of fraudsters whose heterophily ratio are very high. Specifically, over 80% fraudsters have more than 50% heterophilic ratio and over 55% fraudsters have more than 80% heterophily ratio in these relation subgraphs. Consequently, heterophilic connections are widespread in fraud detection, and graph-based fraud detectors should consider both homophilic and heterophilic connections simultaneously.

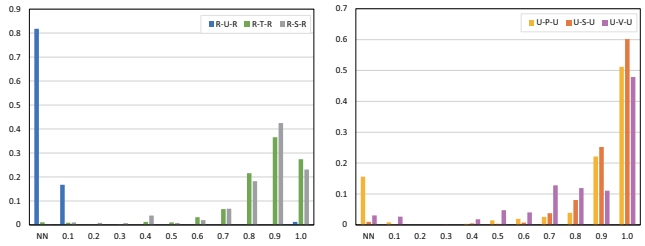
4.3 Performance Comparison (RQ2)

To answer the RQ2, we compare the performance of H^2-FDetector with state-of-the-art methods. The corresponding Recall, AUC, F1-macro and GMean are shown in table 2 and we can make following observations.

First, H^2-FDetector significantly boosts the performance for all metrics on Yelpchi dataset than SOTA results of baselines methods, with 3.76%,12.86%, 14.15% and 14.29% improvement respectively in Recall,F1-macro, AUC and GMean. On Amazon dataset, we observe that H^2-FDetector outperforms other baselines under most of metrics except of F1-macro. On F1-macro, the performance of H^2-FDetector is better than the traditional GNNs and improved GNNs, but worse than the GNN-based fraud methods. The reason is that some benign entities are misjudged as fraudsters. In Amazon dataset, there are very few fraudsters (6.87% in table 1), which leads to most of the adjacent edges of benign entities being homophilic,

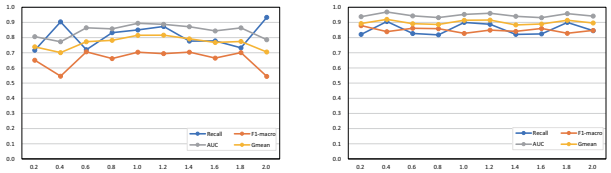
Table 2: Performance Comparison on YelpChi and Amazon.

Method	Dataset	YelpChi				Amazon			
	Metrics	Recall	F1-macro	AUC	GMean	Recall	F1-macro	AUC	GMean
Homophilic	SGC	0.0071	0.4653	0.5243	0.0837	0.4455	0.6652	0.8218	0.6397
	GCN	0.5645	0.4971	0.6112	0.5775	0.6636	0.6577	0.8339	0.7473
	GAT	0.5771	0.4878	0.6091	0.5736	0.4545	0.6948	0.8338	0.6536
Heterophilic	GPRGNN	0.8077	0.5857	0.7952	0.7167	0.8879	0.7285	0.9456	0.8708
	FAGCN	0.8234	0.5448	0.7722	0.6845	0.8394	0.8299	0.9494	0.8887
Fraud Detection	CARE-GNN	0.7052	0.6006	0.7705	0.7052	0.8852	0.8922	0.9416	0.8862
	PC-GNN	0.6721	0.6273	0.7850	0.7088	0.8303	**0.8956**	0.9586	0.9030
Ablation	H^2-FDetector$_{\setminus HP}$	0.8227	0.5488	0.7459	0.6880	0.6758	0.7816	0.9267	0.7983
	H^2-FDetector$_{\setminus PE}$	0.8595	0.6606	0.8764	0.7866	0.8606	0.8178	0.9532	0.8943
H^2-FDetector		**0.8733**	**0.6944**	**0.8877**	**0.8160**	**0.9061**	0.8392	**0.9689**	**0.9203**



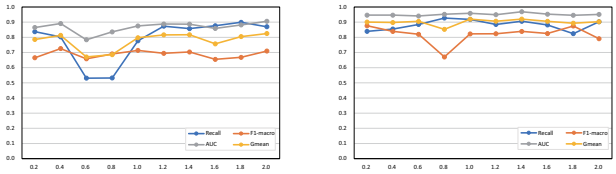
(a) YelpChi (b) Amazon

Figure 3: Heterophilic Evidence. The x-coordinate represents the heterophily ratio of heterophilic edges in the neighborhood of fraud nodes. The y-coordinate represents the proportion of fraud nodes with corresponding heterophily ratio to all fraud nodes."NN" represents the fraud nodes without neighbors.



(a) Hyperparameter γ_1 on YelpChi (b) Hyperparameter γ_1 on Amazon

Figure 4: Sensitive Analysis of Hyperparameter γ_1.



(a) Hyperparameter γ_2 on YelpChi (b) Hyperparameter γ_2 on Amazon

Figure 5: Sensitive Analysis of Hyperparameter γ_2.

but there may also be some heterophilic edges. Our method is sensitive to heterophilic edges, which would increase the probability that the benign entity is judged as a fraudster.

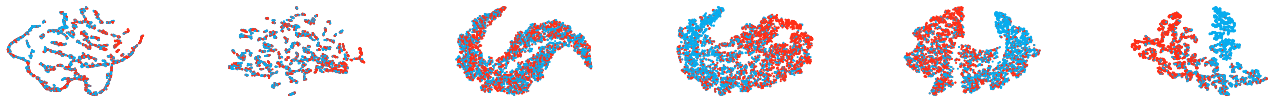
Among the baseline methods, both the traditional GNNs methods (**GCN**, **SGC** and **GAT**) and the graph based fraud detection methods (**CARE-GNN**, **PC-GNN**) regard the fraud graph as homophilic, while the improved GNNs methods **GPRGNN** and **FAGCN** regard the fraud graph as processing the homophilic and heterophilic edges at the same time. From table 2, we can see that on recall, **GPRGNN** and **FAGCN** are significantly better than other methods, which shows that the homophilic and heterophilic connections can find more fraudsters.

Also considering the two homophilic and heterophilic edges, our method is superior to **GPRGNN** and **FAGCN** not only in recall, but also in other evaluation metrics. The reason is that our method can not only perceive homophilic and heterophilic connections and increase the global prototype information of fraudster, but also combine these modules as a whole.

4.4 Ablation study (RQ3)

To answer the RQ3, we construct two ablation models by removing heterophily perception and prototype extraction respectively. The results of two datasets are shown in table 2. We can observe that the performance of two variants decreased significantly, which shows the effectiveness of two modules to H^2-FDetector.

Comparing with GPRGNN and FAGCN, H^2-FDetector$_{\setminus PE}$ substantially outperforms these methods on Yelpchi dataset with respect to all metrics, and slightly better than these methods on Amazon dataset with respect to Recall, AUC and recall. These results suggest that H^2-FDetector$_{\setminus PE}$ can make more accurate identification of fraudsters by perceiving homophilic and heterophilic edges.



(a) GCN (b) GAT (c) CARE-GNN (d) PC-GNN (e) FAGCN (f) H^2-FDetector

Figure 6: Visualization. The red and blue nodes represent fraudsters and benign entities respectively.

4.5 Sensitive Analysis (RQ4)

To answer the RQ4, we evaluate the performance of H^2-FDetector with respect to hyperparameters γ_1 and γ_2. For each hyperparameter, we keep the rest of parameters of model constant, and then we record the corresponding results while varying it from 0.2 to 2 with step size 0.2. The results are shown in figure 4 and figure 5.

From figure 4 (a), we discover that the all metrics are relatively stable and the trends are consistent when γ_1 is around 1. When γ_1 is low, heterophily perception might not be trained sufficiently and may introduce more noise, which lead the AUC, F1-macro and GMean lower. Whereas, introducing heterophily perception loss could include heterophilic neighbors, which provide more information for fraud nodes and is better for recall. Otherwise, if γ_1 is too high, other modules will be difficult to train, which reduces the performance of model. For Amazon dataset, it contains so few fraud nodes that heterophily perception is easy to train. Consequently, γ_1 is more stable on Amazon dataset, which is shown in figure 4 (b).

From figure 5 (a) and (b), the performance of model decreases and then improves as γ_2 gets higher, and finally it remains flat. This phenomenon indicates that the lower γ_2 insufficiently introduce global information while the higher γ_2 cannot introduce more valid information. Consequently, it is reasonable to set the γ_2 around a middle value like 1.2.

4.6 Visualization (RQ5)

To answer the RQ5, we visualize the node embedding of different models, and we take YelpChi dataset for an example. Specifically, based on approximate local aggregation mechanism, we compare H^2-FDetector with GCN, GAT, GPRGNN, CARE-GNN and PC-GNN. Firstly, we learn the node embeddings in a 32-dimensional vector space for different methods, and then we employ the t-SNE [27] to map the 32-dimensional into the 2-dimensional space for visualization. For the efficient and convenient presentation, we show the results of test set and randomly select the same number of negative samples as positive category. The experimental results are shown in Figure 6. From the results, in general, we can make following observations.

First, compared with homophilic-based methods (GCN, GAT, CART-GCN, PC-GNN), H^2-FDetector achieves the separation of fraud nodes from benign nodes obviously. GCN and GAT produce the worst separation due to treating two connection types as only one. Although CARE-GNN and PC-GNN decrease the effect of heterophilic neighbors, the two methods fail to distance the representation of the fraud nodes from that of the benign nodes. Consequently, it is verified that heterophilic neighbor provides information about the difference between classes, which can be used to distance the representation of the fraud nodes from that of the benign nodes.

Second, compared with FAGCN, which consider both homophilic and heterophilic connections as H^2-FDetector, H^2-FDetector produces stronger intra-class cohesion and inter-class segregation. This proves that H^2-FDetector can introduce global information and distinguish the two connections more accurately.

5 RELATED WORK

Heterophily-based Graph Neural Network. Considering the limitation of GNNs based on homophilic assumption, it attracts more attention to design the GNNs considering both homophilic and heterophilic connections. There are two kinds of approaches: 1) The first way is to reduce the heterophilic connections in graph [37, 39]. WRGAT [26] improve the assortativity of graph with graph structure learning (GSL) [2, 8, 14] . 2) The second way combines both homophilic and heterophilic node features by designing new aggregation strategy [40]. GPRGNN [4] adaptively learns the GPR weights so as to optimize node feature with homophilic and heterophilic neighbors. FAGCN [1] use a self-gating mechanism to aggregate both homophilic and heterophilic neighbors.

GNN-based Fraud Detection GNN-based fraud detection is based on homophilic assumption at present[31]. However, there are many heterophilic connections in fraud detection. To ensure the homophilic methods work better, some methods have been proposed[6, 17]. GraphConsis [20] adapts three mechanisms to tackle the inconsistency in fraud detection. Others utilize heterogeneous graph to enrich node information. LIFE [30] conducts embedding learning of both nodes and edges. IHGAT [16] encodes both sequence-like intentions and relationship among transactions for fraud transactions detection. However, the homophilic assumption limits above methods to contain category difference information from heterophilic connections.

6 CONCLUSION

In this paper, we first consider both homophilic and heterophilic connections in fraud detection and propose a Graph Neural Network-based Fraud Detector with Homophilic and Heterophilic Connections (H^2-FDetector for short). Extensive experiments on two benchmark fraud datasets demonstrate the effectiveness of the proposed method.

ACKNOWLEDGMENTS

This work is supported by the Youth Innovation Promotion Association of the Chinese Academy of Sciences (No. 2018192), Strategic Priority Research Program of Chinese Academy of Sciences (Grant

No. XDC02030000), the NSFC (No. 61872360), the CAS Project for Young Scientists in Basic Research (No. YSBR-008). We would like to thank the anonymous reviewers for their valuable comments.

REFERENCES

[1] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 2021. Beyond Low-frequency Information in Graph Convolutional Networks. In *AAAI*. AAAI Press.

[2] Yu Chen, Lingfei Wu, and Mohammed J. Zaki. 2019. Deep Iterative and Adaptive Learning for Graph Neural Networks. *CoRR* abs/1912.07832 (2019). arXiv:1912.07832 http://arxiv.org/abs/1912.07832

[3] Dawei Cheng, Sheng Xiang, Chencheng Shang, Yiyi Zhang, Fangzhou Yang, and Liqing Zhang. 2020. Spatio-temporal attention-based neural network for credit card fraud detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 362–369.

[4] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *International Conference on Learning Representations*. https://openreview.net/forum?id=n6jl7fLxrP

[5] Sarthika Dhawan, Siva Charan Reddy Gangireddy, Shiv Kumar, and Tanmoy Chakraborty. 2019. Spotting Collective Behaviour of Online Frauds in Customer Reviews. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, Sarit Kraus (Ed.). ijcai.org, 245–251. https://doi.org/10.24963/ijcai.2019/35

[6] Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S Yu. 2020. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 315–324.

[7] Yingtong Dou, Guixiang Ma, Philip S Yu, and Sihong Xie. 2020. Robust spammer detection by nash reinforcement learning. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 924–933.

[8] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. 2019. Learning discrete structures for graph neural networks. In *International conference on machine learning*. PMLR, 1972–1982.

[9] Shuaijun Ge, Guixiang Ma, Sihong Xie, and S Yu Philip. 2018. Securing behavior-based opinion spam detection. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 112–117.

[10] Binbin Hu, Zhiqiang Zhang, Jun Zhou, Jingli Fang, Quanhui Jia, Yanming Fang, Quan Yu, and Yuan Qi. 2020. Loan Default Analysis with Multiplex Graph Learning. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2525–2532.

[11] Parisa Kaghazgaran, Majid Alfifi, and James Caverlee. 2019. Wide-ranging review manipulation attacks: Model, empirical study, and countermeasures. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 981–990.

[12] Parisa Kaghazgaran, James Caverlee, and Majid Alfifi. 2017. Behavioral analysis of review fraud: Linking malicious crowdsourcing to amazon and beyond. In *Proceedings of the International AAAI Conference on Web and Social Media*, Vol. 11.

[13] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[14] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. 2018. Adaptive graph convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[15] Wangli Lin, Li Sun, Qiwei Zhong, Can Liu, Jinghua Feng, Xiang Ao, and Hao Yang. 2021. Online Credit Payment Fraud Detection via Structure-Aware Hierarchical Recurrent Neural Network. IJCAI.

[16] Can Liu, Li Sun, Xiang Ao, Jinghua Feng, Qing He, and Hao Yang. 2021. Intention-aware Heterogeneous Graph Attention Networks for Fraud Transactions Detection. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3280–3288.

[17] Yang Liu, Xiang Ao, Zidi Qin, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2021. Pick and Choose: A GNN-based Imbalanced Learning Approach for Fraud Detection. In *Proceedings of the Web Conference 2021*. 3168–3177.

[18] Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, Le Song, and Yuan Qi. 2019. Geniepath: Graph neural networks with adaptive receptive paths. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4424–4431.

[19] Ziqi Liu, Chaochao Chen, Xinxing Yang, Jun Zhou, Xiaolong Li, and Le Song. 2018. Heterogeneous graph neural networks for malicious account detection. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2077–2085.

[20] Zhiwei Liu, Yingtong Dou, Philip S Yu, Yutong Deng, and Hao Peng. 2020. 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*. 1569–1572.

[21] Julian John McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd international conference on World Wide Web*. 897–908.

[22] Shirin Nilizadeh, Hojjat Aghakhani, Eric Gustafson, Christopher Kruegel, and Giovanni Vigna. 2019. Think outside the dataset: Finding fraudulent reviews using cross-dataset analysis. In *The World Wide Web Conference*. 3108–3115.

[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019), 8026–8037.

[24] Shebuti Rayana and Leman Akoglu. 2015. Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining*. 985–994.

[25] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. 2021. Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Zhi-Hua Zhou (Ed.). International Joint Conferences on Artificial Intelligence Organization, 1548–1554. Main Track.

[26] Susheel Suresh, Vinith Budde, Jennifer Neville, Pan Li, and Jianzhu Ma. 2021. Breaking the Limit of Graph Neural Networks by Improving the Assortativity of Graphs with Local Mixing Patterns. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (2021).

[27] Laurens van der Maaten and Geoffrey Hinton. 2008. Visu-alizing data using t-SNE. journal of Machine Learning Research 9. *Nov (2008)* (2008).

[28] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations* (2018). https://openreview.net/forum?id=rJXMpikCZ accepted as poster.

[29] Daixin Wang, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, Shuang Yang, and Yuan Qi. 2019. A semi-supervised graph attentive network for financial fraud detection. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 598–607.

[30] Haishuai Wang, Zhao Li, Peng Zhang, Jiaming Huang, Pengrui Hui, Jian Liao, Ji Zhang, and Jiajun Bu. 2021. Live-Streaming Fraud Detection: A Heterogeneous Graph Neural Network Approach. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3670–3678.

[31] Jianyu Wang, Rui Wen, Chunming Wu, Yu Huang, and Jian Xion. 2019. Fdgars: Fraudster detection via graph convolutional networks in online app review system. In *Companion Proceedings of The 2019 World Wide Web Conference*. 310–316.

[32] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint arXiv:1909.01315* (2019).

[33] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *The World Wide Web Conference*. 2022–2032.

[34] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. PMLR, 6861–6871.

[35] Yichao Wu and Yufeng Liu. 2007. Robust truncated hinge loss support vector machines. *J. Amer. Statist. Assoc.* 102, 479 (2007), 974–983.

[36] Bingbing Xu, Huawei Shen, Bingjie Sun, Rong An, Qi Cao, and Xueqi Cheng. 2021. Towards Consumer Loan Fraud Detection: Graph Neural Networks with Role-Constrained Conditional Random Field. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4537–4545.

[37] Liang Yang, Chuan Wang, Junhua Gu, Xiaochun Cao, and Bingxin Niu. 2021. Why Do Attributes Propagate in Graph Convolutional Neural Networks?. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4590–4598.

[38] Yiming Zhang, Yujie Fan, Yanfang Ye, Liang Zhao, and Chuan Shi. 2019. 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*. 549–558.

[39] Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. 2021. Graph Neural Networks with Heterophily. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 11168–11176.

[40] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. *Advances in Neural Information Processing Systems* 33 (2020).