Alleviating Structural Distribution Shift in Graph Anomaly Detection

Yuan Gao
yuanga@mail.ustc.edu.cn
University of Science and Technology
of China

Xiang Wang*
xiangwang1223@gmail.com
University of Science and Technology
of China

Xiangnan He*
xiangnanhe@gmail.com
University of Science and Technology
of China

Zhenguang Liu
liuzhenguang2008@gmail.com
Zhejiang University

Huamin Feng
oliver_feng@yeah.net
Beijing Electronic Science and
Technology Institute

Yongdong Zhang
zhyd73@ustc.edu.cn
University of Science and Technology
of China

ABSTRACT

Graph anomaly detection (GAD) is a challenging binary classification problem due to its different structural distribution between anomalies and normal nodes — abnormal nodes are a minority, therefore holding high heterophily and low homophily compared to normal nodes. Furthermore, due to various time factors and the annotation preferences of human experts, the heterophily and homophily can change across training and testing data, which is called structural distribution shift (SDS) in this paper. The mainstream methods are built on graph neural networks (GNNs), benefiting the classification of normals from aggregating homophilous neighbors, yet ignoring the SDS issue for anomalies and suffering from poor generalization.

This work solves the problem from a feature view. We observe that the degree of SDS varies between anomalies and normal nodes. Hence to address the issue, the key lies in resisting high heterophily for anomalies meanwhile benefiting the learning of normals from homophily. Since different labels correspond to the difference of critical anomaly features which make great contributions to the GAD, we tease out the anomaly features on which we constrain to mitigate the effect of heterophilous neighbors and make them invariant. However, the prior distribution of anomaly features is dynamic and hard to estimate, we thus devise a prototype vector to infer and update this distribution during training. For normal nodes, we constrain the remaining features to preserve the connectivity of nodes and reinforce the influence of the homophilous neighborhood. We term our proposed framework as *Graph Decomposition Network* (GDN). Extensive experiments are conducted on two benchmark datasets, and the proposed framework achieves a remarkable performance boost in GAD, especially in an SDS environment where

anomalies have largely different structural distribution across training and testing environments. Codes are open-sourced in https://github.com/blacksingular/wsdm_GDN.

CCS CONCEPTS

• **Security and privacy** → **Web application security**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

Graph Neural Networks, Anomaly Detection, Out-of-Distribution

1 INTRODUCTION

Anomalies (*aka.* fraudsters) delineate the abnormal objects that deviate significantly from the normal (*aka.* benigns) [28]. Detecting anomalies has attracted considerable attention in many real-world domains, such as identifying spams in reviews [12], misinformation in social networks [8], and frauds in financial transactions [25]. In general, abnormal and normal objects are interdependent with rich relationships, which can be naturally organized as graphs [12]. Wherein, nodes represent these objects, and edges interpret their relationships. On such structural data, leading methods [12, 25, 49] frame the graph abnormal detection (GAD) problem as the semi-supervised node classification task, where only a fraction of nodes are labeled as the training data, and the remaining nodes form the testing data. To distill the discriminative information for the hidden anomalies, these methods mostly apply graph neural networks (GNNs) [14, 21, 41] that propagate the label-aware signals along with the graph structure.

Here we take a closer look at GAD, especially the structural distribution *w.r.t.* heterophily and homophily. Specifically, **heterophily** [34] indicates the phenomenon that edges connect the nodes from different classes (*i.e.,* the anomaly and normal classes), which contrasts with **homophily** that counts for edges between the same-class nodes. Clearly, for a node, the distribution *w.r.t.* heterophily and homophily shows the label information within its local neighborhood, as shown in Figure 1(a). The distribution difference between anomaly and normal nodes is amplified by GAD's class imbalance nature — that is, anomalies are in the minority subgroups and submerged in the normal nodes, thus easily holding higher heterophily than normals, as the comparison between Figures 1(a) and (d) shows. Hence, it is of importance to differentiate
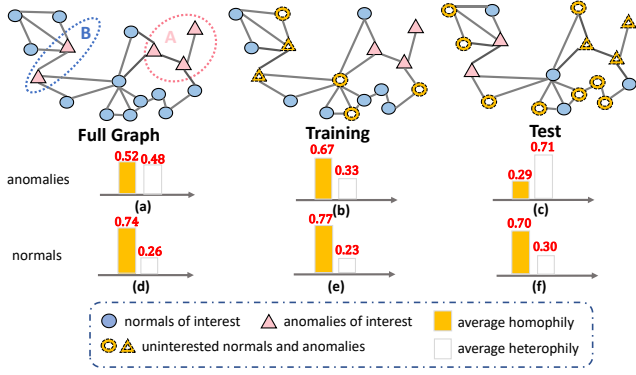
*Corresponding authors

Figure 1: Illustration of SDS in GAD. Cells in the dotted line mean that they are of no interest in the current environment (training or test). The average homophily and heterophily for anomalies and normals are presented.

the structural distributions of anomaly and normal nodes, so as to refine the class-wise signals better.

Furthermore, we find that conducting GAD in a semi-supervised manner naturally faces the challenge of structural distribution shift (SDS) — that is, the structural distribution *w.r.t.* heterophily and homophily can change across the training and testing datasets. For example, as compared between Figures 1(b) and (c), heterophily of anomalies is 0.67 and 0.29 in the training and testing environments, respectively; in contrast, heterophily of normal nodes only is much more stable across two different environments, as Figures 1(e) and (f) show. Here we present several insights into SDS:

- SDS happens in the open environment, which usually includes multiple, different distributions that are collected with various time factors [46], annotation preferences [15]. Hence, robustness to SDS is critical for deploying GAD models in real-world scenarios.
- Distinguishing anomalies from the normal suffer from the shift *w.r.t.* heterophily. Specifically, as most GNN-based GAD models [12, 25, 49] blindly aggregate the neighborhood information, the representations of testing anomalies absorb more information from normal neighbors than that of training anomalies, thereby inundating the crucial cues. Therefore, these GAD models will generalize poorly in the testing anomalies.
- In contrast to anomalies, normals hold more stable distributions across the training and testing sets. Thus, classifying normals can benefit from the class-aware patterns of homophilous neighbors.

However, alleviating SDS in GAD remains largely unexplored, but is the focus of our work. Specifically, most early studies [14, 21, 41] blindly employ GNNs to perform information propagation among nodes, without inspecting the influence of different neighbors. Some follow-up works [12, 25, 49] control the information being propagated by discarding some edges based on the neighbor similarity. Although these studies help mitigate the heterophily gap between anomalies and normals, they leave the SDS issue untouched. Thus, they struggle to fit the testing anomalies with the learned patterns from the training nodes.

In this paper, we argue that the key to alleviating SDS is differentiating structural patterns for anomalies and normals. We assume

that (partial) anomaly features are quite useful for GAD[7]. These features have high variance across anomalies and normal nodes, therefore in which dimensions nodes are more likely to absorb noisy signals from heterophilous neighbors. This observation leads us to the feature disentanglement. Inspired by variable decomposition in stable learning [13, 38], we devise a new framework, Graph Decomposition Network (GDN). Specifically, for anomalies, it tries to identify the anomaly pattern which is made invariant to SDS, so as to reduce the negative influence of heterophily shift; meanwhile, for normals, it attempts to extract the pattern retaining to benefit of homophily. To achieve these two strategies, it resorts to constraints on node features and divides them into two parts: class and surrounding features. Class features are constrained to approach the prior distribution of node features, so as to prevent anomalies from absorbing noisy signals from the neighborhood and influences from heterophily shift; meanwhile, surrounding features preserve the connectivity of two neighboring nodes, to enhance the information of homophilous neighbors. This allows us to identify anomaly features invariant to heterophily shift and capture the local homophily of normals, thus boosting the overall GAD performance.

2 PRELIMINARIES

In this section, we illustrate the task of GAD and the imbalanced heterophily property.

Graph Anomaly Detection. Conventional anomaly detection techniques always consider isolated data instances while ignoring the relationship between instances which carries complementary information [1]. Taking spam review for e-commerce as an example, multiple relations can be established between reviews, *e.g.,* reviews posted by the same user, so as those posted under the same item. In this manner, we reorganize the anomalies and normals as an attributed multi-relation graph, which can be defined as $\mathcal{G} = \{\mathcal{V}, \{\mathcal{E}_r\}, \mathbf{X}\}$. $\mathcal{V}$ denotes a set of anomaly and normal nodes; $\mathcal{E}_r$ stands for edges *w.r.t.* relation $r \in \{1, 2, ..., R\}$, which can be rules, interactions, or shared attributes between nodes [12]; $\mathbf{X}$ is the attribute matrix, each row of which is a d-dimensional vector representing the features of the corresponding node.

GAD has been addressed as a semi-supervised node classification task. Most of the time, anomalies are regarded as positive with label 1, while normal nodes are seen as negative with label 0. The whole graph contains two types of nodes, $\mathcal{V}_{train}$ and $\mathcal{V}_{test}$. $\mathcal{V}_{train}$ are labeled with $\mathbf{Y}_{train}$, while the labels $\mathbf{Y}_{test}$ are inaccessible during training. Formally, leading solutions [12, 25, 49] employ GNN as the predictive model f to achieve small error on predicting the ground truth $\mathbf{Y}_{test}$ for unobserved nodes $\mathcal{V}_{test}$

$$f(\mathcal{G}, \mathbf{Y}_{train}) \rightarrow \hat{\mathbf{Y}}_{test}. \tag{1}$$

Heterophily and homophily. Given a set of nodes along with their labels, the heterophily of a node can be defined as the ratio of the edges connecting the nodes in different classes (*i.e.,* the anomaly and normal classes). For each node, the sum of its heterophily and homophily degree equals 1:

$$X_{hetero}(v) = \frac{1}{|\mathcal{N}(v)|}|\{u : u \in \mathcal{N}(v), y_u \neq y_v\}|,$$

$$X_{homo}(v) = \frac{1}{|\mathcal{N}(v)|}|\{u : u \in \mathcal{N}(v), y_u = y_v\}|, \tag{2}$$

Table 1: Heterophily and homophily in Amazon

	Training		Test	
	Homo	Hetero	Homo	Hetero
Anomalies	0.113	0.887	0.043	0.957
Normal	0.979	0.021	0.976	0.024

where $\mathcal{N}(\cdot)$ stands for the neighborhood of a central node. In GAD, enhanced by the imbalance nature, anomalies have high heterophily, while normal nodes have relatively low heterophily. An example on real dataset is shown in Table 1, from which we observe this phenomenon, as well as the shift *w.r.t.* heterophily and homophily across training and test environments.

3 METHODOLOGY

In this section, we present the details of the proposed framework GDN. We will first introduce the Structural Distribution Shift (SDS) problem in GAD. Then we elaborate on gradient-based invariant feature extraction as well as two constraints that guide the process, respectively. Finally, we present the overview of our proposed model GDN.

3.1 Structural Distribution Shift in GAD

We next formulate the structural distribution shift problem and figure out two major concerns: *how will SDS affect the learning of GNNs and what causes the problem?*

3.1.1 Definition. SDS means the label distribution of the neighborhood is different between p_{train} and p_{test}, which leads to different homophily and heterophily degrees for each node v:

$$p_{train}([X_{hetero}(v_1), X_{homo}(v_1)]) \neq p_{test}([X_{hetero}(v_2), X_{homo}(v_2)])$$
(3)

It is usually caused by human annotation: anomalies with more intra-class edges are easier to be marked out. For easier expression, we denote the joint probability distribution $[X_{hetero}(v), X_{homo}(v)]$ as $\Psi(v)$, then the probability of an instance v being labeled not only depends the node feature x_v but on $\Psi(v)$:

$$p_{train}(\Psi(v), x_v, o) \neq p(x_v, o)$$
(4)

where p is the real distribution of samples, o denotes observed training nodes. The extent of SDS can be defined as:

$$\mathcal{D}[p_{train}(\Psi(v), x_v, o), p(x_v, o)],$$
(5)

where $\mathcal{D}(p(x), g(x)) = \int_{\chi} p(x)\psi(\frac{g(x)}{p(x)})dx$ is a distance function in Csiszár family, which is set as KL divergence in this work.

3.1.2 Effects of Structural Distribution Shift on GAD. GNN-based GAD methods capture the neighborhood pattern and transfer it to unseen nodes, which do not consider the SDS. This distribution inconsistency hinders the model from optimizing the ideal loss function which could definitely harm the model's performance [48]:

$$\mathbb{E}_{v \sim p}[l(g(\Psi(v), x_v))] \neq \mathbb{E}_{(v,o) \sim p_{train}}[l(g(\Psi(v), x_v))|o = 1]$$
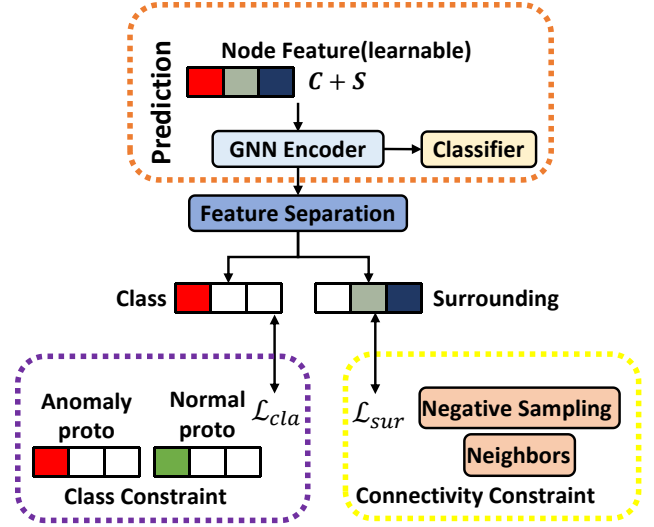(6)

Figure 2: Illustration of GDN. The feature separation module separates the node feature into two sets. Two constraints are leveraged to assist separation. Blank positions in node representation mean they are zero when calculating losses.

where $\mathbb{E}v \sim p[l(g(\Psi(v), x_v))]$ is the expected value of the objective function for GNN-based methods over the distribution of all examples, and the right-hand side expectation is the expected value of loss solely on training (observed) set. For semi-supervised learning, the right-hand side expectation is directly minimized since only nodes with $o = 1$ are available. In GAD, anomalies only occupy the minority of nodes and hold vast normal neighbors, making it a severe SDS environment. More specifically, anomalies, as the detection target, tend to have more anomaly neighbors in the training set, while connecting with more normal nodes in the testing. In an extreme case, all of the anomaly-anomaly edges are contained in the training set, and the neighbor label distribution of anomalies is totally skewed in the test set. Therefore, the ideal loss function can not be directly optimized, GNN classifier fails to capture the real underlying distribution of data samples.

3.1.3 A closer look in SDS. We present here an improved analysis through a closer look at the cause of SDS. First, we need to make an assumption about the probability distribution $P(O_i)$, which denotes whether the node is labeled, *i.e.,* a node is labeled if $O_i = 1$, and $O_i = 0$ otherwise. Following [48], we adopt a biased selection method to construct examples that have the SDS problem between training and testing environments. As discussed above, for each node, the probability of it being labeled is controlled by its neighborhood label similarity:

$$P(O_i = 1) = |\{j|j \in \mathcal{N}_i, y_i = y_j\}|/|\mathcal{N}_i|$$
(7)

In this manner, if the homophily between central node i with its neighborhood is larger, node i will have a larger probability to appear in the training set. This setting is consistent with our cognition of the real world: an anomaly surrounded by more anomalies is easier to be identified.

(a) Anomalies in Amazon

(b) Normals in Amazon
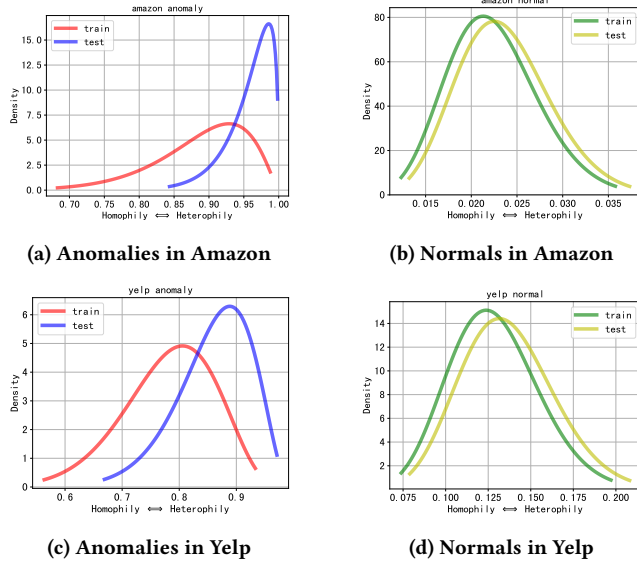
(c) Anomalies in Yelp

(d) Normals in Yelp

Figure 3: Visualization of SDS for different classes of nodes on two real world datasets. On x-axis, left means higher homophily and right means higher heterophily.

Consider one-hop neighbor set $\mathcal{N}(v_i)$ of a given central node v_i, then for binary classification problem, label distribution of each node v_j in $\mathcal{N}(v)$ obeys $\mathbf{y} \sim Bern(p_j)$, where p_j is the probability of v_i and v_j have the same label. Then $\Psi(v)$, *i.e.*, the label distribution of $\mathcal{N}(v)$, is a family of probability distributions whose domain is bounded between 0 and 1, which leads us to Proposition 2.

Proposition 2 The label neighborhood distribution of a given node should obey $\mathbf{Y} \sim Be(\alpha, \beta)$.

With the above proposition, we split two real-world spam-review detection datasets according to this assumption based on which we conduct further analysis. The visualizations of $\Psi(v)$ on training and test are presented in Figure 3. From the figure, we have two observations. (1) Comparing 3a and 3b (3c and 3d), the structural distribution shift on anomalies is apparent, while that on normals is trivial, which indicates that different learning strategies should be adopted for different class nodes. (2) The extent of SDS is positively correlated with the extent of heterophily presented in Table 1, which suggests that heterophily may be the cause of SDS.

3.2 Invariant Feature Extraction

In Section 3.1.3, we inspect the cause of SDS, and suppose that imbalanced heterophily would exacerbate the problem. To alleviate the negative effect of SDS, the key is to identify a pattern expected to be invariant to SDS for anomalies. Intuitively, we want to reduce the neighbor label influence for anomalies. However, in GNNs the information is directly propagated through node features instead of labels, hence we want to bridge the gap between label influence and feature influence. Luckily, the recent method [43] theoretically formulates the relationship between them, and shows the edge weights which serve to aggregate node features also aggregate node labels over its immediate neighbors. From this perspective,

we can mitigate the neighbor label influence for anomalies by reducing feature influence. Furthermore, we assume that (partial) features are useful for detection[7], *i.e.*, the distinguishment of anomalies. These features are quite different across different class nodes, therefore they are more likely to absorb noisy signals from heterophilous neighbors. This observation leads us to the invariant feature extraction.

For anomalies, inspired by variable decomposition [13, 38], we assume that node feature X can be decomposed into *class feature* C and *surrounding feature* S. We hope C inherits most of the node informative characteristics which depict "what the anomaly prototype looks like regardless of heterophily", this set of features will be constrained by prototype (introduced in section 3.3) which prevent the representation from deviating to heterophily neighbor signals. Formally, for variable X_{train}, Y_{train} on training distribution p_{train}:

$$\mathbb{E}[Y_{train}|\Phi(X_{train})] = \mathbb{E}[Y_{train}|C_{train}, S_{train}]$$
$$= \mathbb{E}[Y_{train}|C_{train}] = \mathbb{E}[Y|C], \tag{8}$$

where $\Phi(X_{train})$ is the neighbor feature distribution of training nodes. Note that in GNNs, a node itself is always included in $\Phi(X_{train})$ by adding self-loop, as C inherits most of the node informative characteristics, S is independent to label Y given C. According to the assumption, C is a property of node and is invariant to neighbors, it is unlikely affected by heterophily and SDS, *i.e.*, $\mathbb{E}[Y_{train}|C_{train}] = \mathbb{E}[Y|C]$.

For normals, since they have low heterophily and hence their neighborhood information is more constant and cleaner, we expect that they aggregate neighborhood information which assist their own representation learning. We hope S can capture local structure near normal nodes and summarize "how normal nodes benefit from the low heterophily", a connectivity constraint(introduced in section 3.3) is applied to ensure neighbors share similar S. We next introduce how to appropriately separate features to meet our demand above.

Recall that we aim to extract the most informative C, intuitively, as the spirit of gradient descent, neural importance can be quantified as its absolute score of gradient value *w.r.t.* prediction loss. Following recent works [6, 32], which utilize Grad-CAM [36] or its variants to obtain local contribution to classification, we formulate the contribution of the k-th feature to the anomaly detection at layer l as:

$$\alpha_k^{(l,c)} = \frac{1}{N} \left| \sum_{n=1}^{N} \frac{\partial y^{(c)}}{\partial H_{k,n}^{(l)}} \right|, \tag{9}$$

where $y^{(c)}$ is the predicted probability of ground truth c, H is the hidden layer feature representations and N is the total number of samples. We design a feature selector on the basis of this gradient score, which adaptively teases out class feature C using top-K sampling, and leaves the rest as surrounding feature S. During the selection process, to ensure C and S meet expectations, we next introduce two constraints on the learning objectives.

3.3 Constraints

Towards the goal of invariant feature extractions, we revise the learning objective of GAD by enforcing: 1) *class constraints*: maximize the similarity of C between two nodes if they belong to the

same class, while minimizing it when they have the different labels, and 2) *connectivity constraints*: maximize the similarity of S between two nodes if they are neighbors, otherwise minimize it. We term these two constraints as $\mathcal{L}_{cla}$ and $\mathcal{L}_{sur}$, and introduce them as follows:

$$\mathcal{L}_{cla}(v) = KL(C_v, proto_+) - KL(C_v, proto_-)$$

$$\mathcal{L}_{sur}(v) = \sum_{u \in \mathcal{N}(v)} KL(S_u, S_v) - \sum_{u \notin \mathcal{N}(v)} KL(S_u, S_v)$$

$$\mathcal{L}_{constraint} = \frac{1}{|N|} \sum_{v: y_v=1} \mathcal{L}_{cla}(v) + \mathcal{L}_{sur}(v), \tag{10}$$

where $proto_+$ and $proto_-$ are prior distributions for anomalies and normals respectively, and the distributions depict "what the prototype looks like" for each class; KL is the KL-divergence between two distributions. $\mathcal{L}_{sur}$ is easy to handle, by randomly sampling non-neighbors from the dataset and computing the distances between neighbors and non-neighbors respectively. Now we have transformed the problem into distribution estimation. A prevalent way is to learn a class-specific global context that gives a broad overview of the given class, namely the prototype vector. In our model, we acquire this prototype vector $proto$ adaptively. In every epoch, we register the current prototype $proto^{(e)}$, based on which we calculate the similarity between each node and $proto$. Nodes that deviated from $proto$ from this epoch should have a lower weight in the next update step [5]. Formally, for nodes in each class:

$$s_v^{(e)} = cosine(h_v^{(e)}, proto^{(e-1)}), \quad w_v^{(e)} = \frac{exp(s_v^{(e)}/\tau)}{\sum_{u=1}^{N} exp(s_u^{(e)}/\tau)}$$

$$proto^{(e)} = \sum_{v=1}^{N} w_v \cdot h_v^{(e)} \tag{11}$$

where τ is the temperature parameter to control the smoothness of weights. First, we compute the cosine similarity between each node v and previous prototype vector $proto^{(e-1)}$, whose softmax output is served as weight $w_v^{(e)}$ for each node. Then the prototype $proto^{(e)}$ is updated by aggregating different nodes accordingly on the basis of this contribution. In practice, we initialize $proto^{(0)}$ as the average pooling of the vectors in the given class. The prototype vector serves as a constraint to endow class feature C with resistance to heterophily and SDS.

3.4 Final Loss Function

Figure 2 presents an overview of our proposed model GDN. Following leading solutions, we adopt RGCN [35] as our backbone. For each node v, we define its final embedding as the output of the RGCN at the last layer $z_v = h_v^{(l)}$. And we leverage our constraint on cross-entropy as the final loss function for optimization:

$$\mathcal{L}_{GDN} = \sum_{v \in \mathcal{V}} -log(y_v \cdot \sigma(z_v)) + \lambda exp(\mathcal{L}_{constraint}) \tag{12}$$

where λ is a hyper-parameter to control the balance between two losses. Note that node vectors should locate closer to same-class prototype than other-class one, thus $\mathcal{L}_{cla}$ is always negative; likewise, S of neighbor nodes should look more similar than those of non-neighbor nodes, therefore, $\mathcal{L}_{sur}$ is less than zero; then the

Table 2: Statistics of Datasets

Dataset	#Nodes	#Edges	Relation	#Edges
YelpChi	45954	3846979	R-U-R	49315
			R-S-R	3402743
			R-T-R	573616
Amazon	11944	4398392	U-P-U	175608
			U-S-U	3566479
			U-V-U	1036737

sum of two terms $\mathcal{L}_{constraint}$ is negative, which we transform to the exponential space for the sake of positivity of total loss.

4 EXPERIMENTS

In this section, we conduct experiments on real-world datasets and report the results of our models as well as some state-of-the-art baselines to show the effectiveness of our proposed model. Particularly, we mainly aim to answer the following research questions:

- **RQ1:** How does our model GDN perform compared with SOTA graph anomaly detection methods?
- **RQ2:** Can GDN alleviate the SDS problem on biased dataset?
- **RQ3:** Is the proposed model effective under different hyper-parameter settings?
- **RQ4:** Is GDN framework flexible to various GNN backbones and helpful in enhancing them?

4.1 Experimental Setup

4.1.1 Dataset. Following previous works, We use the Yelp dataset [33] and Amazon dataset [29] to study GNN-based fraud detection problem. The Yelp dataset includes hotel and restaurant reviews filtered and recommended by Yelp. The Amazon dataset includes product reviews under the Musical Instruments category. Similar to [12], we label users with more than 80% helpful votes as benign entities and users with less than 20% helpful votes as fraudulent entities. Both of the datasets are attributed multi-relation graph. Nodes in Yelpchi dataset have 32-dimension features and 3 relations, and nodes in Amazon have 25-dimension features and 3 relations. For exact definition of these relations, we refer the readers to [12, 25]. Table 2 shows the dataset statistics.

4.1.2 Baselines. We are not aware of any similar work addressing SDS. As our focus is GAD, we choose some state-of-the-art methods in the task, some of which utilize the same backbone as us.

- **GCN** [21]: GCN is a traditional graph convolutional network in spectral space.
- **Graph-SAGE** [14]: Compared with GCN, Graph-SAGE samples and aggregates features from a node's local neighborhood, which can generalize to unseen nodes.
- **GAT** [41]: A method that leverages masked self-attentional layers to address the shortcomings of prior graph convolution methods.
- **GraphConsis** [26]: GraphConsis is a heterogeneous graph neural network focusing on tackling context inconsistency, feature inconsistency and relation inconsistency problem.
- **Care-GNN** [12]: Care-GNN is a camouflage-resistant graph neural network that adaptively samples neighbors according

to feature similarity, and the optimal sampling ratio is found through an RL module.

- **FRAUDRE** [49]: This method wants to be dual-resistant to graph inconsistency and imbalance. An additional graph convolution module is leveraged to magnify the difference between normals and anomalies.
- **PC-GNN** [25]: This method consists of two modules "pick" and "choose", and maintain a balanced label frequency around fraudsters by downsampling and upsampling.

4.1.3 Metrics. Since Graph anomaly detection is always a class-imbalanced classification problem, accuracy is not a proper evaluation metric as a high accuracy is still achieved when all of the nodes are classified as normal. Therefore, similar to previous works, we adopt three widely used measures for fair comparison: F1-macro, AUC and GMean.

F1-macro calculates F1-score for every class and finds their unweighted mean. F1-score is the harmonic mean of precision and recall. AUC is the area under the ROC Curve, which depicts the relationship between False Positive Rate (FPR) and True Positive Rate (TPR). Points on the ROC curve with higher AUC scores tend to locate nearer to top-left, thus having lower FPR and higher TPR. GMean is the geometric mean of True Positive Rate (TPR) and True Negative Rate (TNR). For all of the three metrics, the higher scores indicate the higher performance of the approaches.

4.1.4 Implementation Details. The average with standard deviations of 5 runs is reported for all experiments. The backbone of the proposed method is RGCN whose hidden dimension is set to 64. Following previous work, our data splitting ratio is 40%, 20%, and 40% for training, validation, and test set. All of the hyper-parameters are tuned based on the validation set. λ is ranged from {0.01, 0.1, 0.5, 1}, k for yelp is chosen from {6, 12, 18, 24}, while that for amazon is chosen from {5, 10, 15, 20}.

4.2 Comparison Results

To answer **RQ1**, we evaluate the performance of baselines and the proposed method, and the comparison results are reported in Table 3. We implement GCN and Graph-SAGE on our own in Pytorch, and for GraphConsis, CARE-GNN, FRAUDRE, and PC-GNN, we use their provided open-source code to implement them. All of the hyperparameters are set to those reported in their paper if available. We have the following observations:

First of all, CARE-GNN, FRAUDRE, and PC-GNN are three well-designed models for GAD. They are built upon a multi-relation graph, and utilize RGCN as the backbone model as we do. They focus on modifying adjacency matrix to prune noise edges or maintain a balanced neighborhood label frequency, while different from them, we are aiming at learning a powerful and expressive node presentation, which can avoid the high risk of information loss when deleting edges. These three methods are the most competitive baselines so far, and experimental results show that the proposed method consistently outperforms them on all the metrics across two datasets. FRAUDRE performs poorly compared to PC-GNN and CARE-GNN, we think the reason behind this is that FRAUDER deals with the minority class at the end of the topology as a separate module, which may not effectively contribute to the learning
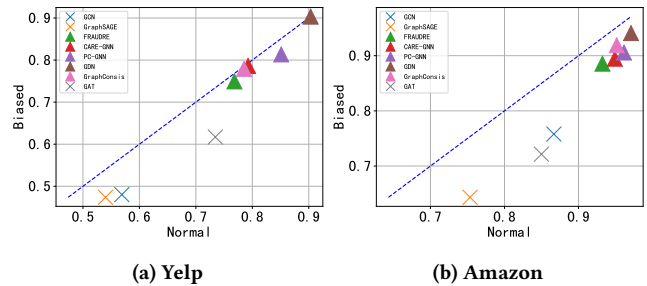

(a) Yelp   **(b) Amazon**

Figure 4: Performance comparison under normal setting and biased setting, the measure is AUC.

process of GNNs. And PC-GNN outperforms CARE-GNN, which is consistent with the conclusion in the paper of PC-GNN.

Secondly, Graph-SAGE is an inductive graph learning method, as it samples a subset of nodes and substantially decreases the difficulty in training on graph for its memory efficiency. However, this sampling technique may also face a high risk for loss of information which leads to performance drop, especially in high imbalanced-heterophily datasets. Therefore, there exists a huge gap between the proposed method GDN and them.

Thirdly, general graph learning methods are evaluated on single-relation graph where all edges are merged. As seen in the table, GAD methods which are built upon multi-relation graph have obvious advantages over GCN, Graph-SAGE and GAT. It demonstrates the effectiveness of treating different relations differently. We think the reason behind is that edges in this manner have a more specific semantic information.

4.3 Out-of-Distribution Evaluation

To answer **RQ2**, we split datasets according to Equation (7). The train-valid-test split ratio remains the same with the normal setting, and note that we treat both train nodes and validation nodes as *observed*, *i.e.,* their neighborhood distributions are the same. We suppose this processing could simulate the real-world better, and help with fair comparison when doing hyper-parameter tuning. We conduct experiments on the proposed method and baselines. From the result reported in Figure 4, we can observe that:

The proposed method outperforms all of the baselines under both biased and normal settings, which again demonstrates the effectiveness of our model. In addition, multi-relation methods marked by triangles perform much better than single-relation methods marked by "X". Next, as compared between Figures 4 (a) and (b), it is obvious that SDS (indicated by the distance from points to the dashed blue line $y = x$) is more severe on Amazon than that on YelpChi, we attribute this difference to the difference in heterophily of anomalies. From Figure 3, we can see the heterophily degree is higher on Amazon than that on Yelp. This finding is consistent with our assumption that SDS is exaggerated by heterophily. Since GDN's performance gap between biased setting and normal setting on both benchmarks is small compared with other GAD methods, our model has the promising ability to alleviate SDS.

Table 3: Performance Results. Best results of all methods are indicated in bold face, and second best results are underlined.

Method	Dataset	YelpChi			Amazon		
	Metric	F1-macro	AUC	GMean	F1-macro	AUC	GMean
General GNNs	GCN	$0.5171_{\pm0.0097}$	$0.5689_{\pm0.0157}$	$0.4541_{\pm0.0946}$	$0.6054_{\pm0.0883}$	$0.8667_{\pm0.0027}$	$0.7638_{\pm0.0343}$
	GraphSAGE	$0.4184_{\pm0.0620}$	$0.5400_{\pm0.0043}$	$0.3207_{\pm0.1560}$	$0.5835_{\pm0.0088}$	$0.7535_{\pm0.0059}$	$0.7037_{\pm0.0076}$
	GAT	$0.5164_{\pm0.0986}$	$0.7403_{\pm0.0242}$	$0.6227_{\pm0.0549}$	$0.6426_{\pm0.0359}$	$0.8499_{\pm0.0014}$	$0.6268_{\pm0.1265}$
GAD models	GraphConsis	$0.6577_{\pm0.0012}$	$0.7853_{\pm0.0033}$	$0.6779_{\pm0.0165}$	$0.7894_{\pm0.0448}$	$0.9516_{\pm0.0005}$	$0.8787_{\pm0.0025}$
	FRAUDRE	$0.5765_{\pm0.0482}$	$0.7683_{\pm0.0176}$	$0.6978_{\pm0.0195}$	$0.8682_{\pm0.0300}$	$0.9299_{\pm0.0035}$	$0.8768_{\pm0.0185}$
	CARE-GNN	$0.6433_{\pm0.0094}$	$0.7925_{\pm0.0292}$	$0.7094_{\pm0.0359}$	$\underline{0.8988_{\pm0.0073}}$	$0.9491_{\pm0.1115}$	$0.8908_{\pm0.0018}$
	PC-GNN	$\underline{0.6933_{\pm0.0253}}$	$\underline{0.8512_{\pm0.0015}}$	$\underline{0.7720_{\pm0.014}}$	$0.8658_{\pm0.0074}$	$\underline{0.9614_{\pm0.0014}}$	$\underline{0.8978_{\pm0.0044}}$
Ours	GDN	$\mathbf{0.7605_{\pm0.006}}$	$\mathbf{0.9034_{\pm0.008}}$	$\mathbf{0.8084_{\pm0.0009}}$	$\mathbf{0.9068_{\pm0.0042}}$	$\mathbf{0.9709_{\pm0.0016}}$	$\mathbf{0.9078_{\pm0.0011}}$
	Improvement	9.69%	6.13%	4.72%	0.89%	0.98%	1.11%

Table 4: LR and LP prediction results with different feature combinations. The measure is AUC.

		C	$C^{'}$	$C + S$	$(C + S)^{'}$
LR	Amazon	0.9069	0.9047	0.9084	0.9062
	Yelp	0.7314	0.7432	0.7545	0.7694
LP	Amazon	0.9277	0.9289	0.9293	0.9298
	Yelp	0.7285	0.7377	0.7418	0.7447

Figure 5: Feature Separation Stability Analysis.

4.4 Feature Separation Analysis

In this section, we will verify the effectiveness of our feature separation module from both the data perspective and model perspective. Inspired by [19], we first visualize the frequency distributions of class features C on 5 different random seeds. Results are reported in Figure 5. In Figure 5(a), we adopt kernel density estimation (KDE) and fit the shape of distribution with Gaussian Kernel; in Figure 5(b), a detailed index constitution of class feature for each seed is displayed. From the figure, we observe that the frequency of selected features on different seeds is around 5, which indicates that the framework always selects the same feature dimension, and our separation process is stable and constant. From model perspective, we aim to verify the assumption that C inherits most of the samples' informative characteristics, and S improves the performance of graph-based methods. Towards this end, we measure the performance of different feature combinations on simple linear classifiers and graph-based classifiers, because we suppose that the simpler the classifier, the better it reflects the feature quality. We train Logistic Regression classifiers and Label Propagation classifiers on original class features (C), regularized class features ($C^{'}$), original features ($C + S$), and regularized features ($(C + S)^{'}$), respectively. Note that LP algorithm reconstructs the graph according to nodes' similarity and is trained on the new graph instead of the original graph. Therefore, LP is less likely to suffer from class heterophily, whose performance is a good measurement of nodes' ability to represent their local structure. Results are displayed in Table 4, and from the table, we have some observations:

Comparing Table 4 with Table 3, the linear model (LR) achieves better performance than general GNN-based methods like GAT, GraphSAGE and GCN. It is consistent with previous work and our assumption that graph-based methods suffer from imbalanced heterophily problem in GAD. Secondly, C has little performance drop on LR compared with $C + S$, suggesting that C has the ability
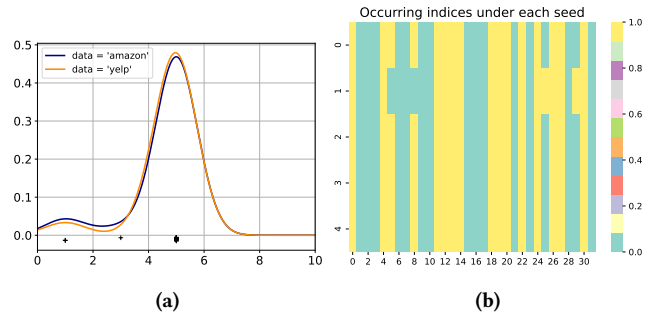
to represent most information contained in the whole feature set. What's more, $C + S$ outperforms C on LP, indicating that including S in the feature set is conducive to the performance of graph-based methods. In addition, $C^{'}$ and $(C + S)^{'}$ achieves better or comparable performance to C and $C + S$, which demonstrates the effectiveness of our regularization terms.

4.5 Hyper-parameter Sensitivity Analysis

To answer **RQ3**, we want to explore the model's sensitivity to the most important parameters λ and k, which control the integration of two losses and the dimensions of C and S when conducting feature separation. The results of three metrics with different λ and k are shown in Figure 6, respectively. We observe that (1) Generally, when λ continues to increase, the performance will first increase then decrease. We suppose a small λ has an inadequate influence on the classification while the classifier may be dominant by the auxiliary regularization when it is too large. (2) Similarly, with the increase of k, the performance tends to first increase then decrease. (3) The performance is stable over a quite large range.

4.6 Flexibility Analysis

To further verify the flexibility of GDN and answer **RQ4**, we leverage feature separation module and regularization terms on several representative GNNs and observe large improvement on both of the datasets. Experiment results are reported in Table 5. From table 5, we can conclude that our proposed method can be easily adapted to general models and enhance their performance.
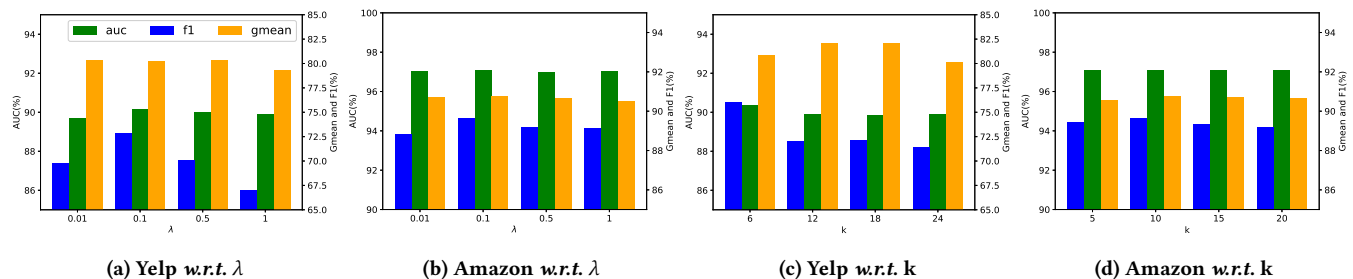
Figure 6: Model performance with different hyper-pamameters on Yelp and Amazon Dataset. Since three metrics are different in scales, we display the charts in a dual-Y style. The left y axis represents AUC, while the right one represents Gmean and F1.

Table 5: Enhancement for other models. The measure is AUC.

	GCN	GCN+GDN	SAGE	SAGE+GDN
Amazon	0.8667	0.8904	0.7535	0.8216
Yelp	0.5689	0.5832	0.5400	0.7467

5 RELATED WORK

In this section, we introduce some representative previous works on graph-based anomaly detection and out-of-distribution learning.

5.1 Graph Anomaly Detection

Graph anomaly detection (GAD) is a task focusing on identifying individual nodes which are obviously deviated from others. Researchers in this area tend to categorize the problem into two branches based on whether the graph data is time-invariant [1, 28]. Hence, we summarize existing GAD works from these two views.

On static attributed graphs, auto-encoder (AE) based methods can not handle the graph data directly. DONE [2] trains two AEs by minimizing the reconstruction error while preserving the homophily between connected nodes. With the advances of GNNs, GNN-based methods [10, 23, 50] have been of focus. GraphRfi [50] explores the possibility of combining anomaly detection with downstream graph tasks. GraphUCB [11] adopts contextual multi-armed bandit technology, and transform graph anomaly detection to a decision-making problem. DCI [45] decouples representation learning and classification with the self-supervised learning task. In real-world, time-variant dynamic graphs are more common. Besides structural information and node attributes, they also contain temporal signals which represent the evolving process of an event. F-FADE [4] uses frequency-factorization to model the time-evolving distributions of frequencies of interactions between node-pairs.

Recent methods realize the importance of incorporating multiple relationships into graph learning [12, 24–26, 42, 44]. FdGars [44] and GraphConsis [26] construct a single homo-graph with multiple relations and leverage GNNs to aggregate neighborhood information. Similarly but differently, Semi-GNN [42], CARE-GNN [12], and PC-GNN [25] construct multiple homo-graphs based on node relation. Semi-GNN and IHGAT [24] employ hierarchical attention mechanism for interpretable prediction, while based on camouflage behaviors and imbalanced problem, CARE-GNN and PC-GNN prune edges adaptively according to neighbor distribution.

5.2 Learning under Out-of-Distribution

In OOD problem, *i.i.d* assumption is violated, hence the classic supervised learning methods cannot be directly applied. Existing algorithms for OOD generalization can be divided into unsupervised representation learning and supervised model learning [39].

Unsupervised representation learning methods are mainly based on VAE. β-VAE [17] introduces an learnable hyperparameter β, balancing latent channel capacity and independence constraints. FactorVAE [20] improves β-VAE by adding a total correlation penalty, and disentangles by encouraging the representation distribution to be factorial and independent. Similarly, some recent works learn variables in a causal graph in an unsupervised fashion. CausalVAE [47] leverages a causal layer that transforms independent exogenous factors into causal endogenous ones. Evidence lower bound is introduced to make the model training easier. DEAR [37] uses a structural causal model (SCM) as the prior for a bidirectional generative model, and a suitable GAN loss is introduced as supervision to train a generator and an encoder jointly with the prior.

Supervised model learning wants to learn a robust model that can be generalized to the unseen target domain. CIAN [22] leverages deep neural network to learn domain-invariance representation with respect to the joint distribution of representation learning function and feature. Graph-DVD [38] decorrelates the stable feature from unstable variables, which is then used for classification. Some works [9, 16, 30, 31] want to connect invariance to causality. These methods are based on ICP [30], which performs a statistical test on whether the invariance assumption is met.

The methods mentioned above tackle the problem well in the euclidean space. However, the OOD problem is under-explored in graph-structured data. SR-GNN [51] and an IRM-based method [46] notices OOD problems in node feature distribution, and a PAC-Bayesian analysis [27] demonstrates non-IID data can affect the performance of subgroups. In this work, we suppose that neighborhood information also suffers from SDS.

the recent study [18] theoretically shows that a node's information gained from the surrounding neighbors is positively correlated to its feature smoothness. Thus removing edges according to feature similarity is at high risk for the loss of the most important neighborhood information.

6 CONCLUSION AND FUTURE WORK

In this work, we explain a novel problem structural distribution shift in graph anomaly detection task. To alleviate SDS, we propose a novel method GDN. GDN separates node features into two parts: one ascertain an invariant pattern for anomalies to describe the anomaly portrait, while the other endows with the ability to absorb rich neighborhood information and benefit from aggregation mechanism under low heterophily for normals.

The method takes the first step to address structural distribution shift in GAD. For future work, there are some research directions worth studying: 1) Integration of feature constraint and edge pruning. Noisy edges are truly harmful, an OOD-guided edge pruning method deserves our attention. Also a mechanism to integrate both algorithms is helpful. 2) Better regularizer. The discovery of a better distance function for regularization terms, such as f-divergence and Wasserstein distance, since KL-divergence has some limitations. 3) Spectral domain. Some GAD works [3, 40] recently published find that anomaly nodes can incur high frequency, addressing SDS in spectral domain is another future direction.

7 ACKNOWLEDGMENTS

REFERENCES

[1] Leman Akoglu, Hanghang Tong, and Danai Koutra. 2015. Graph based anomaly detection and description: a survey. *Data Min. Knowl. Discov.* 29, 3 (2015), 626–688.

[2] Sambaran Bandyopadhyay, Lokesh N, Saley Vishal Vivek, and M. Narasimha Murty. 2020. Outlier Resistant Unsupervised Deep Architectures for Attributed Network Embedding. In *WSDM*. 25–33.

[3] Ziwei Chai, Siqi You, Yang Yang, Shiliang Pu, Jiarong Xu, Haoyang Cai, and Weihao Jiang. 2022. Can Abnormality be Detected by Graph Neural Networks?. In *IJCAI*.

[4] Yen-Yu Chang, Pan Li, Rok Sosic, MH Afifi, Marco Schweighauser, and Jure Leskovec. 2021. F-fade: Frequency factorization for anomaly detection in edge streams. In *WSDM*. 589–597.

[5] Bo Chen, Jing Zhang, Xiaokang Zhang, Yuxiao Dong, Jian Song, Peng Zhang, Kaibo Xu, Evgeny Kharlamov, and Jie Tang. 2021. GCCAD: Graph Contrastive Coding for Anomaly Detection. *CoRR* abs/2108.07516 (2021).

[6] Long Chen, Xin Yan, Jun Xiao, Hanwang Zhang, Shiliang Pu, and Yueting Zhuang. 2020. Counterfactual samples synthesizing for robust visual question answering. In *CVPR*. 10800–10809.

[7] Zhixian Chen, Tengfei Ma, and Yang Wang. 2022. When Does A Spectral Graph Neural Network Fail in Node Classification? *CoRR* abs/2202.07902 (2022).

[8] Lu Cheng, Ruocheng Guo, Kai Shu, and Huan Liu. 2021. Causal understanding of fake news dissemination on social media. In *KDD*. 148–157.

[9] Gregory C Chow. 1960. Tests of equality between sets of coefficients in two linear regressions. *Econometrica: Journal of the Econometric Society* (1960), 591–605.

[10] Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. 2019. Deep anomaly detection on attributed networks. In *SDM*. 594–602.

[11] Kaize Ding, Jundong Li, and Huan Liu. 2019. Interactive anomaly detection on attributed networks. In *WSDM*. 357–365.

[12] 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 *CIKM*. 315–324.

[13] Shaohua Fan, Xiao Wang, Chuan Shi, Kun Kuang, Nian Liu, and Bai Wang. 2022. Debiased Graph Neural Networks with Agnostic Label Selection Bias. *TNNLS* (2022).

[14] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*.

[15] Fengxiang He, Tongliang Liu, Geoffrey I Webb, and Dacheng Tao. 2018. Instance-dependent pu learning by bayesian optimal relabeling. *arXiv preprint arXiv:1808.02180* (2018).

[16] Christina Heinze-Deml, Jonas Peters, and Nicolai Meinshausen. 2018. Invariant causal prediction for nonlinear models. *Journal of Causal Inference* 6, 2 (2018).

[17] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*.

[18] Yifan Hou, Jian Zhang, James Cheng, Kaili Ma, Richard TB Ma, Hongzhi Chen, and Ming-Chang Yang. 2020. Measuring and improving the use of graph information in graph neural networks. In *ICLR*.

[19] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. 2022. GraphLIME: Local Interpretable Model Explanations for Graph Neural Networks. *TKDE* (2022).

[20] Hyunjik Kim and Andriy Mnih. 2018. Disentangling by factorising. In *ICML*. 2649–2658.

[21] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[22] Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. 2018. Deep domain generalization via conditional invariant adversarial networks. In *ECCV*. 624–639.

[23] Jiongqian Liang, Peter Jacobs, Jiankai Sun, and Srinivasan Parthasarathy. 2018. Semi-supervised embedding in attributed networks with outliers. In *SDM*. 153–161.

[24] 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 *KDD*. 3280–3288.

[25] 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 *WWW*. 3168–3177.

[26] 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 *SIGIR*. 1569–1572.

[27] Jiaqi Ma, Junwei Deng, and Qiaozhu Mei. 2021. Subgroup generalization and fairness of graph neural networks. In *NIPS*.

[28] Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Chuan Zhou, Quan Z Sheng, Hui Xiong, and Leman Akoglu. 2021. A comprehensive survey on graph anomaly detection with deep learning. *TKDE* (2021).

[29] Julian John McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *WWW*. 897–908.

[30] Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. 2016. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* (2016), 947–1012.

[31] Niklas Pfister, Peter Bühlmann, and Jonas Peters. 2019. Invariant causal prediction for sequential data. *J. Amer. Statist. Assoc.* 114, 527 (2019), 1264–1276.

[32] Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. 2019. Explainability methods for graph convolutional neural networks. In *CVPR*. 10772–10781.

[33] Shebuti Rayana and Leman Akoglu. 2015. Collective opinion spam detection: Bridging review networks and metadata. In *KDD*. 985–994.

[34] Everett M Rogers and Dilip K Bhowmik. 1970. Homophily-heterophily: Relational concepts for communication research. *Public opinion quarterly* 34, 4 (1970).

[35] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*. 593–607.

[36] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*. 618–626.

[37] Xinwei Shen, Furui Liu, Hanze Dong, Qing Lian, Zhitang Chen, and Tong Zhang. 2020. Disentangled Generative Causal Representation Learning. *CoRR* abs/2010.02637 (2020).

[38] Zheyan Shen, Peng Cui, Jiashuo Liu, Tong Zhang, Bo Li, and Zhitang Chen. 2020. Stable learning via differentiated variable decorrelation. In *KDD*. 2185–2193.

[39] Zheyan Shen, Jiashuo Liu, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. 2021. Towards Out-Of-Distribution Generalization: A Survey. *CoRR* abs/2108.13624 (2021).

[40] Jianheng Tang, Jiajin Li, Ziqi Gao, and Jia Li. 2022. Rethinking Graph Neural Networks for Anomaly Detection. In *ICML*. 21076–21089.

[41] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[42] 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 *ICDM*. 598–607.

[43] Hongwei Wang and Jure Leskovec. 2020. Unifying graph convolutional neural networks and label propagation. *arXiv preprint arXiv:2002.06755* (2020).

[44] 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 *WWW (Companion Volume)*. 310–316.

[45] Yanling Wang, Jing Zhang, Shasha Guo, Hongzhi Yin, Cuiping Li, and Hong Chen. 2021. Decoupling representation learning and classification for gnn-based anomaly detection. In *SIGIR*. 1239–1248.

[46] Qitian Wu, Hengrui Zhang, Junchi Yan, and David Wipf. 2022. Towards Distribution Shift of Node-Level Prediction on Graphs: An Invariance Perspective. In *ICLR*.

[47] Mengyue Yang, Furui Liu, Zhitang Chen, Xinwei Shen, Jianye Hao, and Jun Wang. 2021. CausalVAE: disentangled representation learning via neural structural causal models. In *CVPR*. 9593–9602.

[48] Bianca Zadrozny. 2004. Learning and evaluating classifiers under sample selection bias. In *ICML*. 114.

[49] Ge Zhang, Jia Wu, Jian Yang, Amin Beheshti, Shan Xue, Chuan Zhou, and Quan Z Sheng. 2021. FRAUDRE: Fraud Detection Dual-Resistant to Graph Inconsistency and Imbalance. In *ICDM*. 867–876.

[50] Shijie Zhang, Hongzhi Yin, Tong Chen, Quoc Viet Hung Nguyen, Zi Huang, and Lizhen Cui. 2020. GCN-Based User Representation Learning for Unifying Robust Recommendation and Fraudster Detection. In *SIGIR*. ACM, 689–698.

[51] Qi Zhu, Natalia Ponomareva, Jiawei Han, and Bryan Perozzi. 2021. Shift-robust gnns: Overcoming the limitations of localized graph training data. In *NIPS*.