GraphDE: A Generative Framework for Debiased Learning and Out-of-Distribution Detection on Graphs

Zenan Li, Qitian Wu, Fan Nie, Junchi Yan*
Department of Computer Science and Engineering and MoE Key Lab of Artificial Intelligence
Shanghai Jiao Tong University
{emiyali, echo740, youluo2001, yanjunchi}@sjtu.edu.cn

Abstract

Despite the remarkable success of graph neural networks (GNNs) for graph representation learning, they are generally built on the (unreliable) i.i.d. assumption across training and testing data. However, real-world graph data are universally comprised of outliers in training set and out-of-distribution (OOD) testing samples from unseen domains, which solicits effective models for i) debiased learning and ii) OOD detection, towards general trustworthy purpose. In this paper, we first mathematically formulate the two challenging problems for graph data and take an initiative on tackling them under a unified probabilistic model. Specifically, we model the graph generative process to characterize the distribution shifts of graph data together with an additionally introduced latent environment variable as an indicator. We then define a variational distribution, i.e., a recognition model, to infer the environment during training of GNN. By instantiating the generative models as two-component mixtures, we derive a tractable learning objective and theoretically justify that the model can i) automatically identify and down-weight outliers in the training procedure, and ii) induce an effective OOD detector simultaneously. Experiments on diverse datasets with different types of OOD data prove that our model consistently outperforms strong baselines for both debiasing and OOD detection tasks. The source code has been made publicly available at https://github.com/Emiyalzn/GraphDE.

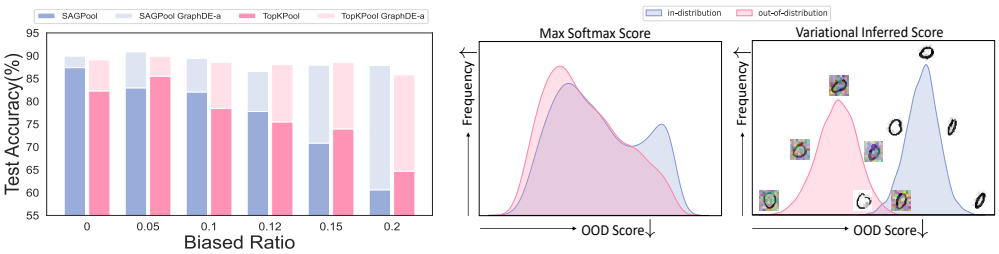
1 Introduction

For processing ubiquitous graph data, graph neural networks (GNNs) [18, 43] have emerged as effective approaches for graph representation learning [12], which in turn has spanned a wide range of applications, from drug discovery [23] to image classification [4].

Despite their success, existing GNNs are generally built on the assumption that both training and testing data are independently sampled from the identical distribution (i.i.d.), which often does not hold [45, 46]. Real-world training data is often shown universally mixed with outliers [30, 7], which specifically may incur large gradient steps in the wrong direction [38], or be overfitted by the model [3], thereby hurting models' performance. Besides, testing samples from unseen domains are also unavoidable when the model is deployed in the wild [2, 20], which may easily be classified into a wrong class with a high probability [13]. Inevitably, graph data also involve these two common problems. As illustrated in Fig. 1, we indeed find that vanilla GNNs perform poorly when encountering large portion of outliers and out-of-distribution (OOD) testing samples, which solicits effective models for i) debiased learning and ii) OOD detection, towards general trustworthy purpose.

*Junchi Yan is the corresponding author who is also with Shanghai AI Laboratory.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).



(a) Debiasing results on SPMotif

(b) Detection results on MNIST-75sp

Figure 1: Vanilla GNNs' performance degenerates drastically with a larger number of outliers in the training dataset. They also assign high softmax scores to OOD data, which may raise trustworthy concerns. In comparison, our proposed framework can boost performance for both tasks notably. In (a), GraphDE-a is plugged in the backbone; In (b), the variational inferred score is derived from GraphDE's recognition model. We plot MNIST images at corresponding OOD scores for intuition.

However, these two problems still remain less explored for graph data. Generally, identifying graph outliers (or OOD samples) is inherently hard as the non-Euclidean graph data can own a large number of node features and additional structural knowledge. The bias may exist in certain node features, local and global graph structures [29], as well as the underlying relations between features, structures, and labels [38], preventing trivial applications of traditional machine learning methods [25, 6] from identifying outliers on graph data.

We argue that training outliers and OOD testing samples can be conceptually unified as they both represent data drawn from distributions other than the major in-distribution (ID) part of training data. However, they are inherently different as the training outliers also catch label-related distribution shifts, e.g. noisy labels [1], while OOD testing samples are solely determined by their features [14] since we cannot access their ground-truth labels. This difference also explains why many existing works tend to tackle the two problems independently.

In light of the intrinsic connection between these two types of data, we present GraphDE, a probabilistic generative framework for **Graph DE**biased learning and OOD **DE**tection. Specifically, we model the generative process to characterize the distribution shifts of graph data, along with an environment variable as an indicator. Utilizing variational inference [50], we propose to infer the environment variable during the training procedure of GNN. Our framework consists of three modules: the recognition model that infers the environment variable based on the input graph data; the structure estimation model by which we extract an OOD detector on testing dataset; the classification model which is our target GNN. Following the idea of data resampling [38, 37], we model the latter two modules by two-component mixtures, with one component for ID data, and the other component robustifying the model by isolating outliers. Based on this, we derive a tractable learning objective for our model and theoretically justify its effectiveness for the target tasks. **The contributions are:**

- We focus on two critical but largely unexplored problems for graph data: debiased learning and OOD detection, find their fundamental correlations and define them under a probabilistic framework.
- We propose GraphDE with a novel learning objective and theoretically justify that: 1) it can identify and down-weight outliers during training; 2) it can provide an OOD detector on the testing set; 3) its constituent modules are optimized in a mutually-promoting manner.
- We evaluate GraphDE with different GNN backbones on diverse datasets of different OOD types, where GraphDE achieves consistent performance improvements over the baselines. Specifically, as shown in Fig. 1, for the debiasing task, it yields up to 14.6% accuracy improvement on SPMotif; for the OOD detection task, it outperforms the strongest baseline by 9.31% on MNIST-75sp.

2 Problem Formulation and Related Works

In typical graph learning problems, given a set of labeled graphs $\mathcal{D}^{tr} = \{(G_i, y_i)\}_{i=1}^{N^{tr}}$, we need to train a model $f_\theta : G \to \hat{y}$ to predict the labels on the testing dataset $\mathcal{D}^{te} = \{G_i\}_{i=1}^{N^{te}}$. Specifically, the input graph $G = (A, X)$ is composed of an adjacency matrix $A = \{a_{vu} | v, u \in V\}$ and node features

$X = \{x_v | v \in V\}$, where V is the node set. Define $\mathbf{G}$ as a random variable of the input graph and $\mathbf{y}$ as a random variable of the graph label, the data-generating process can then be characterized as $p(\mathbf{G}, \mathbf{y}|\mathbf{e}) = p(\mathbf{G}|\mathbf{e})p(\mathbf{y}|\mathbf{G}, \mathbf{e})$ where we introduce an environment variable $\mathbf{e}$ as a potential unknown indicator for data from particular distributions.

The biased data (we use the form interchangeable to outliers) in the training set and out-of-distribution (OOD) data in the testing set can both be treated as data drawn from different distributions to the distribution generating the majority of the training set and testing data of our interests, i.e., the in-distribution (ID) data so-called. Hence, we can model them by a binary environment variable $\mathbf{e}$, with $\mathbf{e} = 1$ denoting the ID portion and $\mathbf{e} = 0$ for the biased/OOD portion. Then the training and testing datasets can be split into $\mathcal{D}^{tr} = \mathcal{D}^{tr}_{in} \cup \mathcal{D}^{tr}_{out}$ and $\mathcal{D}^{te} = \mathcal{D}^{te}_{in} \cup \mathcal{D}^{te}_{out}$ accordingly. Note that i) the partitions of training/testing data are only for illustration purpose and are strictly unknown in practice; ii) the testing data and distribution are also strictly unknown during the training process; iii) here $\mathbf{e}$ is mainly used for distinguishing the ID part with the OOD counterpart of datasets. In fact, the OOD part ($\mathbf{e} = 0$) may consist of data from multiple distributions [2].

Denote the loss on sample (G, y) by $L(y, f_\theta(G))$, e.g., cross-entropy loss for classification problems. Then the standard empirical risk minimization (ERM) objective can be written as

$$\min_\theta \mathbb{E}_{e\sim p(\mathbf{e})}\Big[\mathbb{E}_{(G,y)\sim p(\mathbf{G},\mathbf{y}|\mathbf{e}=e)}[L(y, f_\theta(G))]\Big]. \tag{1}$$

This objective works based on the i.i.d. assumption across training and testing data. As we aim at promoting the ID test accuracy (i.e. the performance on $\mathcal{D}^{te}_{in}$), however, the updating gradient by training data drawn from $p(\mathbf{G}, \mathbf{y}|\mathbf{e} = 0)$ will hurt model performance drastically [3]. On the other hand, for testing data from unseen distributions $\mathcal{D}^{te}_{out}$, the model would incline to produce imprecise results as $p^{tr}(\mathbf{e}) \neq p^{te}_{out}(\mathbf{e})$, which is undesired and should be avoided for the trustworthy purpose. We next formulate the two problems of our interests.

Task I: Debiased learning for training data. The target is to mitigate outliers' negative effects during training, thereby enhancing the performance on ID testing data. We modify the ERM objective:

$$\begin{aligned} &\min_\theta \mathbb{E}_{e\sim p(\mathbf{e})}\Big[\mathbb{E}_{(G,y)\sim p(\mathbf{G},\mathbf{y}|\mathbf{e}=e)}[L(y, f_\theta(G)|e)]\Big], \\ &\text{where } L(y, f_\theta(G)|e) = \begin{cases} L(y, f_\theta(G)), & \text{if } e = 1, \\ L_0, & \text{if } e = 0. \end{cases} \end{aligned} \tag{2}$$

where the loss function for biased training data is fixed as L_0, so the debiasing objective is equivalent to minimize $\mathbb{E}_{(G,y)\sim p(\mathbf{G},\mathbf{y}|\mathbf{e}=1)}[L(y, f_\theta(G))]$, i.e., the training loss for the ID portion $\mathcal{D}^{tr}_{in}$.

Task II: OOD detection for testing data. We aim to obtain an OOD detector g composed of a trained model $f_\theta : G \to \hat{y}$, a scoring function s and a threshold τ:

$$g(G; \tau, s, f_\theta) = \begin{cases} 0 \text{ (OOD)}, & \text{if } s(G; f_\theta) \leq \tau, \\ 1 \text{ (ID)}, & \text{if } s(G; f_\theta) > \tau. \end{cases} \tag{3}$$

As we are willing to induce an OOD detector during the training procedure of GraphDE, we leverage the trained model f_θ from task I and what we need is to find the optimal score function s^* that can best separate out ID and OOD data (e.g. maximize the AUROC).

Related works on debiased learning. A series of strategies have been proposed to debias the training dataset in general machine learning [38, 5] and computer vision [30, 7]. Many works propose to preprocess the dataset by filtering out the outliers [5, 25]. However, these methods cannot capture label-related distribution variations such as noisy labels [1] and also cannot be easily adapted to deal with graph data. Another popular strategy is data resampling [38, 37], which relies on the training loss to identify and down-weight outliers in the training set. However, these methods only capture bias in the mapping from features to labels, but ignore the variation in feature space. For the first time to our best knowledge, we formally define and deal with the graph debiased learning problem.

Related works on OOD detection. A trustworthy learning system should not only produce accurate predictions for ID data, but also distinguish and reject OOD examples without further prediction.

[2] Since we are not interested in the specific OOD distributions, we model them by a unified distribution $p(\mathbf{G}, \mathbf{y}|\mathbf{e} = 0)$, which is enough to identify the OOD samples as we will justify later.

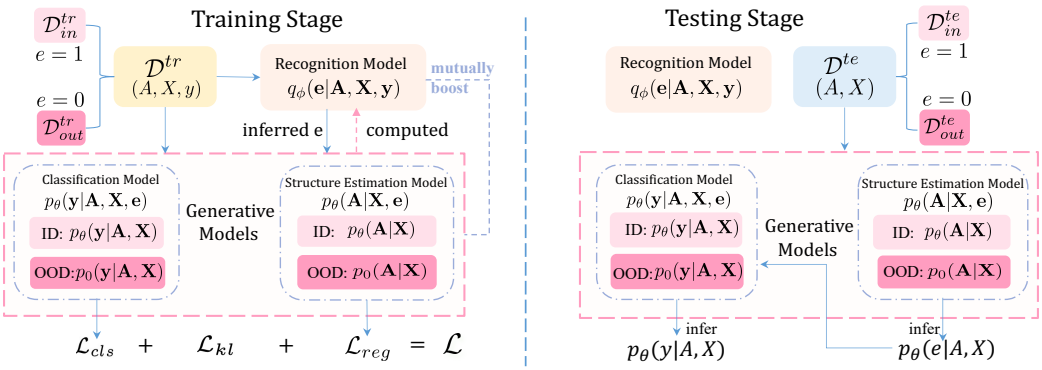


Figure 2: The proposed GraphDE framework. **Left:** During the training stage, the training data is fed into the recognition model to infer the environment variable, then fed into the generative models to compute the total loss for training (we omit the source of $\mathcal{L}_{kl}$ in the figure for simplicity). **Right:** In the testing stage, the testing data is fed into the structure estimation model for OOD detection, then fed into the classification model to compute the class probabilities.

Deep learning based OOD detection has achieved promising progress in vision [27, 13]. Besides, node and edge-level anomaly detection on graphs are also widely studied, some of which are based on classical network science [26], while others utilize the power of GNNs [36, 9]. In comparison, only limited works [52, 29] attempt to solve graph-level OOD detection. Specifically, OCGIN [52] designs a one-class GNN objective to distinguish between ID and OOD data. GLocalKD [29] utilizes joint random distillation of graph and node representations to learn rich global and local pattern information of graph-structured data. However, both of them need to be trained individually for the detection task. In this paper, we extract an OOD detector during the training procedure of the target GNN, while outperforming all the baselines on 7 out of 9 benchmarks. Appendix B gives more comprehensive description on related works on GNNs and Trustworthy GNNs.

3 Methodology

3.1 Probabilistic Generative Learning for Debiased Learning and OOD Detection

To improve the ID test accuracy, we want to make the model more robust by automatically identifying potential outliers during training. Thus, we take a generative perspective on the graph data and adopt variational inference to infer the environment variable during training.

Generative Process. We model the joint distribution $p(\mathbf{X}, \mathbf{y}, \mathbf{A}, \mathbf{e})$ to capture graph-level distribution shifts. In line with the homophily principle from random graph theory [33], i.e., linking probability depends on some inherent properties of nodes, we assume that the graph structure is generated based on the node features and the environment variable. The graph labels are then generated based on the graph structure, node features, and the environment. We also assume $p(\mathbf{e})$ and $p(\mathbf{X})$ to be independent distributions for environments and node features, respectively. We do not specify the form of $p(\mathbf{X})$, since everything will be conditioned on $\mathbf{X}$ in this paper, similar to [28, 44]. We have:

$$p(\mathbf{X}, \mathbf{y}, \mathbf{A}, \mathbf{e}) = p(\mathbf{e})p(\mathbf{X})p(\mathbf{A}|\mathbf{X}, \mathbf{e})p(\mathbf{y}|\mathbf{X}, \mathbf{A}, \mathbf{e}), \tag{4}$$

where the generative models $p(\mathbf{A}|\mathbf{X}, \mathbf{e})$ and $p(\mathbf{y}|\mathbf{X}, \mathbf{A}, \mathbf{e})$ could be modeled by some flexible parametric distributions $p_\theta(\mathbf{A}|\mathbf{X}, \mathbf{e})$ and $p_\theta(\mathbf{y}|\mathbf{X}, \mathbf{A}, \mathbf{e})$ with parameter θ. These two distributions will later be instantiated as the structure estimation model and classification model, respectively. We define the generative models with two-component mixtures. For example, $p_\theta(\mathbf{A}|\mathbf{X}, \mathbf{e})$ is composed of an ID component $p_\theta(\mathbf{A}|\mathbf{X})$, and an OOD component $p_0(\mathbf{A}|\mathbf{X})$ (the subscript 0 implies a fixed distribution). The environment variable $\mathbf{e} \in \{0, 1\}$ indicates whether the graph should be modeled by the ID ($\mathbf{e} = 1$) or the OOD ($\mathbf{e} = 0$) component. Specifically, these two terms are decomposed as:

$$\begin{aligned} p_\theta(\mathbf{A}|\mathbf{X}, \mathbf{e}) &= p_\theta(\mathbf{A}|\mathbf{X})^{\mathbf{e}} p_0(\mathbf{A}|\mathbf{X})^{1-\mathbf{e}}, \\ p_\theta(\mathbf{y}|\mathbf{X}, \mathbf{A}, \mathbf{e}) &= p_\theta(\mathbf{y}|\mathbf{X}, \mathbf{A})^{\mathbf{e}} p_0(\mathbf{y}|\mathbf{X}, \mathbf{A})^{1-\mathbf{e}}. \end{aligned} \tag{5}$$

Recognition Model. As the variables $\mathbf{X}$, $\mathbf{A}$ and $\mathbf{y}$ are all observed during training, what we need is to infer the sample-wise environment variable $\mathbf{e}$ conditional on these observed variables. Thus, we resort to variational inference [50] and introduce a recognition model $q_\phi(\mathbf{e}|\mathbf{A}, \mathbf{X}, \mathbf{y})$ parameterized with ϕ to approximate the true posterior $p_\theta(\mathbf{e}|\mathbf{A}, \mathbf{X}, \mathbf{y})$.

Learning Objective. We train the model parameters θ and ϕ by optimizing the Evidence Lower BOund (ELBO) of the observed data tuple (A, X, y):

$$\begin{aligned}\log p_\theta(A, y|X) \geq & \log p_\theta(A, y|X) - D_{\mathrm{KL}}[q_\phi(\mathbf{e}|A, X, y)||p_\theta(\mathbf{e}|A, X, y)] \\ = & \mathbb{E}_{q_\phi(\mathbf{e}|A,X,y)}[\log(p_\theta(A|X, \mathbf{e})p_\theta(y|X, A, \mathbf{e}))] \\ & - D_{\mathrm{KL}}[q_\phi(\mathbf{e}|A, X, y)||p(\mathbf{e})] = \mathcal{L}_{\mathrm{ELBO}}.\end{aligned} \tag{6}$$

Eq. 6 shows that by maximizing the ELBO, we are maximizing the log-likelihood of observed data $\log p_\theta(A, y|X)$ while minimizing the KL divergence between the recognition model and the true posterior $p_\theta(\mathbf{e}|A, X, y)$. Finally, the learning objective for the training dataset can be written as:

$$\begin{aligned}-\mathcal{L} = & \frac{1}{N^{tr}} \sum_{i=1}^{N^{tr}} \underbrace{\mathbb{E}_{q_\phi(\mathbf{e}_i|A_i,X_i,y_i)} \left[\mathbf{e}_i \log p_\theta(y_i|X_i, A_i) + (1 - \mathbf{e}_i) \log p_0(y_i|X_i, A_i)\right]}_{\textit{Classification Loss } \mathcal{L}_{cls}} \\ & + \frac{1}{N^{tr}} \sum_{i=1}^{N^{tr}} \underbrace{\mathbb{E}_{q_\phi(\mathbf{e}_i|A_i,X_i,y_i)} \left[\mathbf{e}_i \log p_\theta(A_i|X_i) + (1 - \mathbf{e}_i) \log p_0(A_i|X_i)\right]}_{\textit{Structure Regularization Loss } \mathcal{L}_{reg}} \\ & - \frac{1}{N^{tr}} \sum_{i=1}^{N^{tr}} \underbrace{D_{\mathrm{KL}}\left[q_\phi(\mathbf{e}_i|A_i, X_i, y_i)||p(\mathbf{e})\right]}_{\textit{KL Loss } \mathcal{L}_{kl}}.\end{aligned} \tag{7}$$

Justification of GraphDE. We provide theoretical analysis here to reveal the rationale of GraphDE. Without loss of generality, we do not take account into the KL loss here for simplicity. The proofs of the propositions are postponed to Appendix A.

Proposition 1. *1) The learning objective for GraphDE is in a re-weighted form when $q_\phi(\mathbf{e}|\mathbf{A}, \mathbf{X}, \mathbf{y})$ is instantiated as a Bernoulli distribution, with $q_\phi(\mathbf{e}_i = 1|A_i, X_i, y_i)$ acting as a weight for the i-th sample; 2) Given the ideal recognition model q^*_ϕ that gives $q^*_\phi(\mathbf{e} = 1|(A, X, y) \in \mathcal{D}_{in}) = 1$ and $q^*_\phi(\mathbf{e} = 1|(A, X, y) \in \mathcal{D}_{out}) = 0$, the generative models can learn to best fit the ID data.*

Prop. 1 reveals how the inferred posterior takes effect on the learning objective. Typically, if $q_\phi(\mathbf{e}|\mathbf{A}, \mathbf{X}, \mathbf{y})$ can assign a higher probability to ID data, our model can thereby learn to ignore the negative effects of biased training data and better fit the target distribution (i.e. ID).

Proposition 2. *1) Assuming the generative models fit to the ID data, i.e. $p_\theta(A|X \in \mathcal{D}_{in}) \geq p_\theta(A|X \in \mathcal{D}_{out})$ and $p_\theta(y|(A, X) \in \mathcal{D}_{in}) \geq p_\theta(y|(A, X) \in \mathcal{D}_{out})$, the recognition model will learn to predict $q_\phi(\mathbf{e} = 1|(A, X, y) \in \mathcal{D}_{in}) \geq q_\phi(\mathbf{e} = 1|(A, X, y) \in \mathcal{D}_{out})$; 2) Given optimal generative models that best fit the ID data and perform randomly on outliers, there exists a recognition model q^*_ϕ which yields the minimal objective while ideally predict the environment variable.*

Prop. 2 proves GraphDE's effectiveness from the recognition model's perspective. To summarize, the above two propositions show that $q_\phi(\mathbf{e}|\mathbf{A}, \mathbf{X}, \mathbf{y})$, $p_\theta(\mathbf{y}|\mathbf{A}, \mathbf{X})$ and $p_\theta(\mathbf{A}|\mathbf{X})$ can **mutually promote each other** during training. As deep networks tend to learn simple shared (i.e. ID) patterns first [3], this positive feedback can guide GraphDE to identify outliers in early epochs and thereby learn better representations of ID data. Besides, the propositions also suggest that given either of the recognition model or the generative model optimal, the other can be trained to be optimal, which suggests the existence of the global optimum. Based on these, we can further understand the implications of the three loss terms in Eq. 7: i) $\mathcal{L}_{cls}$ is the re-weighted version of the original classification loss; ii) $\mathcal{L}_{reg}$ acts as graph regularization and will impact the gradient of $q_\phi(\mathbf{e}|\mathbf{A}, \mathbf{X}, \mathbf{y})$, weighting ID data higher than the outliers; iii) the third term $\mathcal{L}_{kl}$ is the KL divergence between the learned posterior and the fixed prior distribution, ensuring the posterior not go too complex.

Inference on Testing Dataset. For new data, we need to infer the posterior conditional distribution of $\mathbf{e}$ and $\mathbf{y}$ given $(\mathbf{A}, \mathbf{X})$, i.e. $p_\theta(\mathbf{e}|\mathbf{A}, \mathbf{X})$ and $p_\theta(\mathbf{y}|\mathbf{A}, \mathbf{X})$, respectively. To compute $p_\theta(\mathbf{e}|\mathbf{A}, \mathbf{X})$ is

just to do **OOD detection**. For binary $\mathbf{e}$, we can compute its posterior analytically:

$$p_\theta(\mathbf{e}|\mathbf{A},\mathbf{X}) = \frac{p_\theta(\mathbf{e},\mathbf{A},\mathbf{X})}{\sum_{\mathbf{e}} p_\theta(\mathbf{e},\mathbf{A},\mathbf{X})} = \frac{p(\mathbf{e})p(\mathbf{X})p_\theta(\mathbf{A}|\mathbf{X},\mathbf{e})}{\sum_{\mathbf{e}} p(\mathbf{e})p(\mathbf{X})p_\theta(\mathbf{A}|\mathbf{X},\mathbf{e})}. \tag{8}$$

The posterior can be used as the score function s for the OOD detector, once we have the well-trained $p_\theta(\mathbf{A}|\mathbf{X},\mathbf{e})$. Subsequently, we can calculate $p_\theta(\mathbf{y}|\mathbf{A},\mathbf{X})$ by the total probability theorem:

$$p_\theta(\mathbf{y}|\mathbf{A},\mathbf{X}) = \sum_{\mathbf{e}} p_\theta(\mathbf{y}|\mathbf{A},\mathbf{X},\mathbf{e})p_\theta(\mathbf{e}|\mathbf{A},\mathbf{X}), \tag{9}$$

where $p_\theta(\mathbf{y}|\mathbf{A},\mathbf{X},\mathbf{e})$ can also be obtained after training.

3.2 GraphDE Instantiations

We proceed to specify the distributions $p_\theta(\mathbf{A}|\mathbf{X},\mathbf{e})$, $p_\theta(\mathbf{y}|\mathbf{X},\mathbf{A},\mathbf{e})$, $p(\mathbf{e})$ and $q_\phi(\mathbf{e}|\mathbf{A},\mathbf{X},\mathbf{y})$. We assume $p(\mathbf{e}) = \text{Bernoulli}(\alpha)$ where $\alpha \in [0,1]$ represents our prior belief about the cleanliness of the dataset. A higher α implies a larger portion of ID data in $\mathcal{D}^{tr}$. It is fixed as a scalar during training.

3.2.1 Instantiations of the Recognition Model

We propose two variants for the recognition model $q_\phi(\mathbf{e}|\mathbf{A},\mathbf{X},\mathbf{y})$. In the first way, we directly assign a learnable scalar for each sample in the training dataset, i.e., $q_\phi(\mathbf{e}_i|A_i,X_i,y_i) = \text{Bernoulli}(\alpha_i)$, where $\alpha_i \in [0,1]$ is a trainable parameter jointly optimized with generative models through the ELBO in Eq. 7. We call this variant GraphDE-v(ariational) as we optimize directly over α_i. In the second way, notice that $\mathbf{e}$ is a binary variable by definition, we can compute the posterior analytically by

$$\begin{aligned} q_\phi(\mathbf{e}|\mathbf{A},\mathbf{X},\mathbf{y}) \doteq p_\theta(\mathbf{e}|\mathbf{A},\mathbf{X},\mathbf{y}) &= \frac{p_\theta(\mathbf{X},\mathbf{y},\mathbf{A},\mathbf{e})}{\sum_{\mathbf{e}} p_\theta(\mathbf{X},\mathbf{y},\mathbf{A},\mathbf{e})} \\ &= \frac{p(\mathbf{e})p(\mathbf{X})p_\theta(\mathbf{A}|\mathbf{X},\mathbf{e})p_\theta(\mathbf{y}|\mathbf{X},\mathbf{A},\mathbf{e})}{\sum_{\mathbf{e}} p(\mathbf{e})p(\mathbf{X})p_\theta(\mathbf{A}|\mathbf{X},\mathbf{e})p_\theta(\mathbf{y}|\mathbf{X},\mathbf{A},\mathbf{e})}. \end{aligned} \tag{10}$$

We call this variant GraphDE-a(nalytical) and the optimization procedure in this setting is essentially an EM algorithm [31]: in the E-step, we evaluate the posterior distribution of the latent variable $\mathbf{e}$ by Eq. 10; in the M-step, we optimize the model parameters θ using gradient descent w.r.t. the objective in Eq. 7. In comparison, GraphDE-v runs more efficiently as it does not need to calculate the posterior, while GraphDE-a can give a tight approximation $\log p_\theta(A,y|X) = \mathcal{L}_{\text{ELBO}}$. We compare the space/time complexity of the algorithms in Appendix F.3.

3.2.2 Instantiations of the Structure Estimation Model

Recall that $p_\theta(\mathbf{A}|\mathbf{X},\mathbf{e})$ is defined by a two-component mixture (Eq. 5), we need to specify $p_\theta(\mathbf{A}|\mathbf{X})$ and $p_0(\mathbf{A}|\mathbf{X})$ separately for the structure estimation model. We follow the common practice and assume that the edges are conditionally independent [28]. Then the conditional probability of $\mathbf{A}$ can be further factorized as $p_\theta(\mathbf{A}|\mathbf{X}) = \prod_{v,u\in V} p_\theta(\mathbf{a}_{vu}|\mathbf{X})$.

We next consider two versions for $p_\theta(\mathbf{a}_{vu}|\mathbf{X})$. First, we adopt the latent space models (LSM) [15] where the probability of $\mathbf{a}_{vu}$ only depends on the representations of nodes v and u:

$$p_\theta(\mathbf{a}_{vu} = 1|\mathbf{x}_v,\mathbf{x}_u) = \sigma([(\boldsymbol{U}\mathbf{x}_v)^\top, (\boldsymbol{U}\mathbf{x}_u)^\top]\boldsymbol{w}). \tag{11}$$

Eq. 11 can be viewed as a logistic regression model where $\sigma(\cdot)$ denotes the Sigmoid function, $\boldsymbol{U}$ is a linear transformation matrix, and $\boldsymbol{w}$ is a learnable vector.

Second, we propose a variant of LSM referred as CosLSM which utilizes the cosine similarity function to measure the similarity between the node representation pair $(\mathbf{x}_v,\mathbf{x}_u)$:

$$p_\theta(\mathbf{a}_{vu} = 1|\mathbf{x}_v,\mathbf{x}_u) = \frac{1}{m}\sum_{i=1}^{m} \frac{\cos(\boldsymbol{U}_i\mathbf{x}_v, \boldsymbol{U}_i\mathbf{x}_u) + 1}{2}, \tag{12}$$

where we harness multi-head [42] linear transformations $\{\boldsymbol{U}_i\}_{i=1}^m$ for better capacity. Note that if we train the structure estimation model completely based on the observed edges, the model will yield

trivial solutions which predict all edges' probabilities as 1. To avoid this issue, we randomly sample one disconnected node pair labeled as $a_{vu} = 0$ for each existing edge during training.

Finally, for $p_0(\mathbf{a}|\mathbf{X})$, we simply fix it as $p_0(\mathbf{a}_{vu} = 1|\mathbf{X}) = 1/2$. Although more complex distributions can be used for improving the capacity, our experiment results empirically show that this simple distribution is enough for a wide range of debiasing and detection tasks on graphs.

3.2.3 Instantiations of the Classification Model

We next specify $p_\theta(\mathbf{y}|\mathbf{A}, \mathbf{X})$ and $p_0(\mathbf{y}|\mathbf{A}, \mathbf{X})$ for the classification model. Similar to the structure estimation model, we fix $p_0(\mathbf{y}|\mathbf{A}, \mathbf{X}) = K^{-1}$ where K denotes the total number of classes. Typically, a GNN takes $(\mathbf{A}, \mathbf{X})$ as input and outputs the probability of $\mathbf{y}$. So for $p_\theta(\mathbf{y}|\mathbf{A}, \mathbf{X})$, we adopt two types of GNN models: the global pooling models [18, 43] and hierarchical pooling models [21, 19]. Denote node v's neighborhoods by $\mathcal{N}_v$, specifically, common GNN convolutional layers execute recursive feature propagation along G:

$$\mathbf{x}_v^{(l)} = \sigma\left(\boldsymbol{W}^{(l)}\mathrm{AGG}^{(l)}\left(\{\mathbf{x}_u^{(l-1)}|u \in \mathcal{N}_v \cup \{v\}\}\right)\right),\ \mathbf{X}^{(0)} = \mathbf{X}, \tag{13}$$

where $\boldsymbol{W}^{(l)}$ denotes weight matrix in the l-th layer and $\mathrm{AGG}^{(l)}$ is a permutation-invariant function that aggregates node representations. In hierarchical pooling GNNs, each convolutional layer is followed by a pooling layer to filter out nodes for the next convolution operation:

$$\mathbf{X}^{(l)} = \mathbf{X}_{\text{idx}}^{(l)},\ \mathbf{A}^{(l)} = \mathbf{A}_{\text{idx,idx}}^{(l)}, \tag{14}$$

where *idx* is the selected node indices obtained by operations such as selecting the top-k score (calculated by a learnable function) nodes. Finally, for both types of GNNs, a global pooling layer (e.g. max or mean operation over node features) is adopted to extract a graph-level representation $\mathbf{h}$ to be fed into the classifier Φ whose probability is obtained by an output softmax layer:

$$p_\theta(\mathbf{y}|\mathbf{A}, \mathbf{X}) = \mathrm{Softmax}(\Phi(\mathbf{h})), \mathbf{h} = \mathrm{GlobalPool}(\mathbf{X}^{(n)}), \tag{15}$$

where n denotes the total number of convolutional layers.

4 Experiments

In this section, we conduct extensive experiments to answer the following questions:

- **Q1:** How effective is GraphDE in debiased learning and OOD detection?
- **Q2:** How do the composed modules and hyperparameters impact GraphDE's performance? Besides, how does GraphDE take effect during the training procedure?

4.1 Experiment Setup

Datasets. One synthetic and three real-world datasets of different OOD types are used for debiasing and OOD detection tasks. Here we briefly introduce them, leaving details of statistics to Appendix E.

- **Spurious-Motif** [49] (SPMotif) is a synthetic dataset in which each graph is composed of one base (denoted by S) and one motif (denoted by C). The graph label y is solely determined by the motif C. To simulate distribution shifts, we manually construct spurious correlations between S and y. We use datasets with **different spurious correlations** to represent ID and OOD data, respectively. This dataset is unsuitable for typical OOD detection task, as the distribution shift is dominated by label information, opposed to that the OOD detectors focus on detecting input graphs with abnormal structures or node features.
- **MNIST-75sp** [19] converts the MNIST images into super-pixel graphs with at most 75 nodes. For OOD data, we add Gaussian noise to **node features** to simulate distribution shift.
- **Collab** [48] is a social network derived from 3 public collaboration datasets. We use the **graph size** to simulate distribution shift, with graphs of no more than 45 nodes denoted as ID data.
- **DrugOOD** [16] aims to benchmark the OOD generalization performance in the field of AI-aided drug discovery. We use the provided dataset curator to generate a dataset with **different scaffolds**, and treat the largest 6000 molecules as ID data.

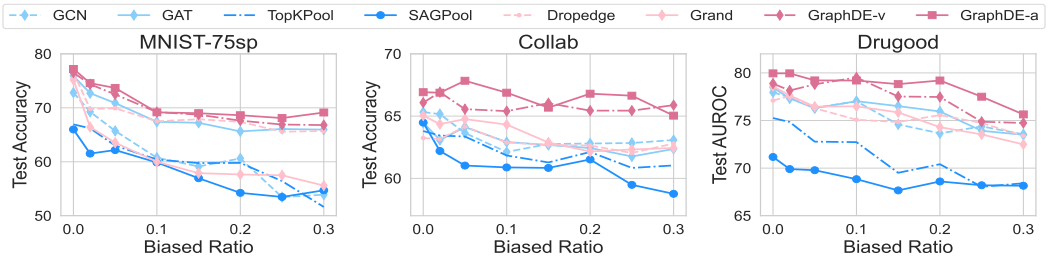


Figure 3: Debiasing results on real-world datasets. The test mean accuracy (or AUROC for DrugOOD) on the ID dataset is plotted. GAT is adopted as the backbone.

Table 1: Debiasing results on SPMotif. It records the test mean accuracy and standard deviation on the ID dataset. The biased ratio is fixed as 10% for fair comparison. Best results are in bold.

Model	Backbone	DropEdge	GRAND	GraphDE-v	GraphDE-a
GCN [18]	68.67±5.04	65.59±3.16	73.92±3.37	75.10±3.36	**76.59±1.92**
GAT [43]	65.02±3.31	62.81±0.64	65.27±3.90	66.88±1.30	**67.52±1.84**
TopKPool [19]	78.43±4.93	82.32±3.85	83.25±3.89	85.11±4.42	**88.54±1.72**
SAGPool [21]	83.57±4.15	85.72±3.15	86.71±3.49	88.96±2.66	**89.41±2.87**

Note that during the debiased training stage of GraphDE, we need to mixed a certain number of OOD data into the training dataset. To prevent potential data leakage [14] during the evaluation of OOD detection, the OOD testing dataset should be drawn from a different distribution w.r.t. the OOD data mixed into the training dataset. We denote these two sources of OOD data as the OOD testing dataset and the OOD mixed dataset, respectively. We leave more details of dataset splits to Appendix E.

GNN backbones. We adopt two types of GNN backbones for debiasing and OOD detection tasks.

- **Global pooling backbones.** We adopt two popular GNNs: GCN [18] and GAT [43]. To extract the global graph representation for graph classification, we use a global mean pooling or global max pooling layer after the GNNs to summarize the node representations.
- **Hierarchical pooling backbones.** This architecture is comprised of blocks each of which consists of a graph convolutional layer and a pooling layer. Typically, TopKPool [19] selects local parts of input graph for propagation by a learnable score function; SAGPool [21] uses the self-attention mechanism to calculate attention scores, and retain important nodes for propagation.

Debiasing settings. For the debiasing task, we mix the training set with certain ratio of outliers and compare models' performance on ID testing data. We evaluate GraphDE's debiasing performance on the four datasets. Besides, As far as we know, there are little existing works concerning debiased learning on graph data. For demonstrating the empirical superiority of our approach, we adopt two recently proposed methods that claim to enhance the general-purpose robustness of GNNs as competitors: [3]: DropEdge [39] randomly removes a number of edges from the input graph in each training epoch, and GRAND [11] uses a random propagation strategy to perform graph augmentation. We study the negative effect of outliers by varying the **biased ratio** ($|\mathcal{D}^{tr}_{out}|/|\mathcal{D}^{tr}|$) of training data.

OOD detection settings. We evaluate GraphDE's OOD detection performance on three real-world datasets (with equal number of ID and OOD testing samples), using three widely used metrics [14]: area under the receiver operating characteristic curve (*AUROC*), area under the precision-recall curve (*AUPR*), and the false positive rate at 95% true positive rate (*FPR95*). Besides, we consider three types of baselines: first, we use the backbones' max softmax score (MSP) [13] as a vanilla baseline; second, we adopt two-stage graph kernel baselines in [52] (specifically, we adopt WL [40] and PK [34] kernels, LOF [5] and OCSVM [6] detectors); finally, we use OCGIN [52] and GLocalKD [29] as two deep learning based detection baselines.

[3]Notably, they can also be integrated with GraphDE to achieve even better debiasing performance.

Table 2: Detection results on real-world datasets. The biased ratio is fixed as 30% for fair comparison. We report the mean and standard deviation for all the detectors, except the two-stage models, which are invariant to different random seeds. We use the GraphDE-a variant as the OOD detector.

OOD Detector	MNIST-75sp			Collab			DrugOOD		
	AUROC↑	AUPR↑	FPR95↓	AUROC↑	AUPR↑	FPR95↓	AUROC↑	AUPR↑	FPR95↓
MSP [13]	62.37±2.96	60.71±1.83	88.60±2.71	51.37±4.24	53.19±3.68	91.00±2.41	54.57±9.18	52.43±6.85	90.76±4.95
WL [40]+OCSVM [6]	75.35	60.72	32.75	64.61	60.39	64.80	66.84	72.41	81.72
WL [40]+LOF [5]	61.62	57.22	94.20	67.72	62.64	81.40	56.21	49.33	80.80
PK [34]+OCSVM [6]	72.26	59.95	47.80	64.57	62.19	70.60	66.08	61.02	82.60
PK [34]+LOF [5]	61.19	58.51	92.55	64.25	58.76	91.20	57.40	51.42	88.20
OCGIN [52]	65.07±2.55	60.13±2.45	77.39±5.55	70.48±2.72	71.77±1.84	86.70±0.71	68.39±4.77	66.05±5.11	82.80±7.50
GLocalKD [29]	86.22±0.78	87.59±0.87	25.72±5.90	71.82±0.47	**72.91±0.24**	70.04±1.32	63.42±0.60	58.03±0.64	**70.28±1.83**
GraphDE	**95.53±3.63**	**94.78±5.09**	**19.24±9.33**	**72.15±2.27**	71.86±2.54	**64.40±0.41**	**69.15±1.11**	**67.40±0.51**	80.30±0.33

4.2 Main Results (Q1)

We train and evaluate each model under the settings as described in the last section (more hyperparameter and training/testing details in Appendix D). All the experiments are run over 5 different random seeds. More experimental results can be found in Appendix F. We make the following observations:

Debiasing results. Tab. 1 reports GraphDE's debiasing performance on SPMotif. In this setting, we treat the robust GNN baselines and two variants of our GraphDE as plug-in modules and apply them to all the four backbones for comparison. The results show that GraphDE consistently (for all the backbones) outperforms the baselines by a large margin along with low variance. Specifically, for the TopKPool backbone, GraphDE-a surpasses the backbone by 10.11% and the strongest baseline (GRAND) by 5.29%. Such improvements strongly suggest that GraphDE can achieve better debiasing performance for various backbones. Besides, the results also show that the GraphDE-a generally outperforms GraphDE-v, as the EM algorithm can give a tight approximation for the log-likelihood.

Fig. 3 illustrates GraphDE's performance on three real-world datasets. In this setting, we mainly study the debiasing performance w.r.t. different biased ratios, so the backbone for robust GNN baselines and GraphDE are all fixed as GAT. As shown in the figure, GraphDE consistently outperforms baselines on the three datasets for all the biased ratios. The backbones' performance drops drastically as the biased ratio gets higher, and the robust GNN baselines can mitigate this effect to some extend. In comparison, GraphDE performs much more stable. This advantage is more pronounced when the biased ratio is large, e.g. GraphDE surpasses all the other baselines by nearly 5% on Collab when the biased ratio is 0.2. An interesting observation is that GraphDE can also achieve a better test accuracy when the biased ratio is 0, implying that the pure training set also violates the i.i.d. assumption.

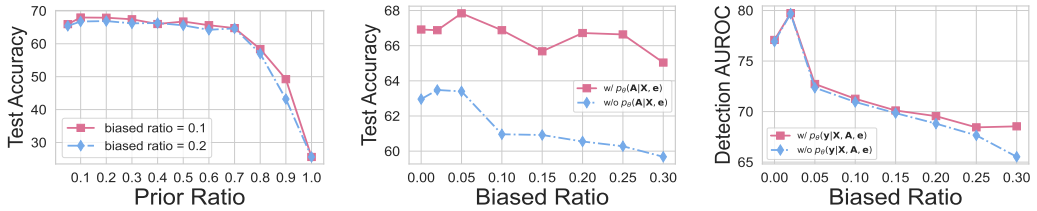
Detection results. Tab. 2 reports GraphDE's detection performance on three real-world datasets. As can be seen from the table, the max softmax score (MSP) achieves low detection performance. It suggests that vanilla GNNs are poorly calibrated [35] and it is of critical need to develop new graph-level OOD detectors. The results show that GraphDE performs the best on 7 out of 9 benchmarks. Specifically, on MNIST-75sp, GraphDE surpasses the strongest baseline by approximately 10% for AUROC, while also pulls down the false positive rate (FPR95).

4.3 Further Study (Q2)

Impact of prior ratio. We select different prior ratios $1 - p(\mathbf{e})$ to study its impact on debiasing performance. As shown in the Fig. 4(a), the ID test accuracy doesn't fluctuate much from 0.05 to 0.7, but drops drastically when prior ratio goes higher than 0.7. Besides, we also find that the test accuracy peaks at prior ratio 0.1 for biased ratio 0.1, and 0.2 for biased ratio 0.2, exactly when $\mathcal{L}_{kl}$ becomes 0.

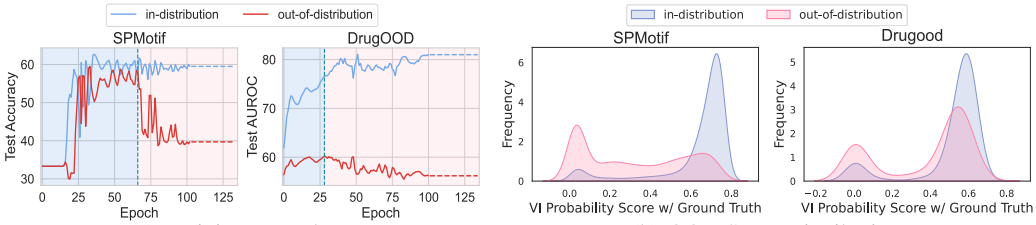
Ablation on $\mathcal{L}_{reg}$. We remove $\mathcal{L}_{reg}$ in the objective to study the structure estimation model's impact on debiasing performance. As illustrated in Fig. 4(b), we can find that the test accuracy drops obviously at all biased ratios. Specifically, The test accuracy drops over 6% at biased ratio 0.25. The results prove the structure estimation model's capability of down-weighting outliers in training data.

Ablation on $\mathcal{L}_{cls}$. We remove $\mathcal{L}_{cls}$ to study the influence of the classification model on the OOD detection performance. As shown in Fig. 4(c), we plot the detection AUROC on the testing dataset w.r.t. different biased ratios. The figure illustrates that the AUROC generally drops as the biased ratio gets higher, since the outliers will hurt the fitting performance of the structure estimation model. Typically, we can see that the blue line (without $\mathcal{L}_{cls}$) is consistently lower than the red line, with the



(a) Prior Ratio (b) Structure Estimation Model (c) Classification Model

Figure 4: Sensitivity analysis and ablation studies (on Collab). We use the GraphDE-a variant.



(a) Training Procedure (b) OOD Score Distribution

Figure 5: Explanatory experiments. We use the GraphDE-a variant and the biased ratio is set as 0.2.

gap getting larger as the biased ratio goes higher. Specifically, the performance gap is approximately 3% when biased ratio is 0.3. The results prove that the classification model can help down-weight training outliers, thereby promoting the detection performance of the structure estimation model.

Training procedure. We plot GraphDE's test accuracy after every training epoch on ID and OOD (i.i.d. to training outliers) datasets. As shown in Fig. 5(a), the ID test accuracy simply fluctuates, rises, and converges. In comparison, the trend of OOD test accuracy can be divided into two phases. As separated by the green dashed line, we can see the red line first increases then decreases. Generally, in the increasing phase, GraphDE is optimized to better fit the outliers in the training data. However, in the decreasing phase, as the outliers are down-weighted by a large extend, GraphDE focuses on fitting ID data. This transition prevents overfitting of outliers and promotes the ID test accuracy.

Visualization of inferred probability. We visualize the recognition model's inferred probability $q_\phi(e = 1|A, X, y)$ on the training dataset. As shown in Fig. 5(b), ID data is assigned with a higher probability on both datasets. In particular, most outliers are assigned with probability at around 0 on SPMotif. This directly indicates that the outliers are down-weighted during the training procedure.

5 Conclusion and Outlook

In this paper, we have provided a viable approach to the problem of debiased learning and OOD detection over graph data. Specifically, a generative process is modeled to capture distribution shifts of graph data. By introducing a variational recognition model to infer the environment variable and two-component mixed generative models, the learning objective for GraphDE is derived, which can identify and down-weight outliers during training, as well as inducing an effective OOD detector on new data. While our framework is mainly designed for graph data where both the debiased learning and OOD detection remain relatively under-explored, it can be potentially generalized to other data formats like images or texts, which we leave for future studies.

Acknowledgement

This work was partly supported by National Key Research and Development Program of China (2020AAA0107600), National Natural Science Foundation of China (61972250, 72061127003), and Shanghai Municipal Science and Technology (Major) Project (2021SHZDZX0102, 22511105100).

References

[1] G. Algan and I. Ulusoy. Label noise types and their effects on deep learning. *CoRR*, abs/2003.10471, 2020.

[2] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[3] D. Arpit, S. Jastrzębski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, et al. A closer look at memorization in deep networks. In *International conference on machine learning*, pages 233–242. PMLR, 2017.

[4] P. H. Avelar, A. R. Tavares, T. L. da Silveira, C. R. Jung, and L. C. Lamb. Superpixel image classification with graph attention networks. In *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 203–209. IEEE, 2020.

[5] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[6] Y. Chen, X. S. Zhou, and T. S. Huang. One-class svm for learning in image retrieval. In *Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205)*, volume 1, pages 34–37. IEEE, 2001.

[7] T.-J. Chin, Z. Cai, and F. Neumann. Robust fitting in computer vision: Easy or hard? In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 701–716, 2018.

[8] E. Dai, W. jIN, H. Liu, and S. Wang. Towards robust graph neural networks for noisy graphs with sparse labels, 2022.

[9] D. Duan, L. Tong, Y. Li, J. Lu, L. Shi, and C. Zhang. Aane: Anomaly aware network embedding for anomalous link detection. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 1002–1007. IEEE, 2020.

[10] F. Errica, M. Podda, D. Bacciu, and A. Micheli. A fair comparison of graph neural networks for graph classification. *arXiv preprint arXiv:1912.09893*, 2019.

[11] W. Feng, J. Zhang, Y. Dong, Y. Han, H. Luan, Q. Xu, Q. Yang, E. Kharlamov, and J. Tang. Graph random neural network for semi-supervised learning on graphs, 2020.

[12] W. L. Hamilton, R. Ying, and J. Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[13] D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. 2016.

[14] D. Hendrycks, M. Mazeika, and T. Dietterich. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018.

[15] P. D. Hoff, A. E. Raftery, and M. S. Handcock. Latent space approaches to social network analysis. *Journal of the american Statistical association*, 97(460):1090–1098, 2002.

[16] Y. Ji, L. Zhang, J. Wu, B. Wu, L.-K. Huang, T. Xu, Y. Rong, L. Li, J. Ren, D. Xue, et al. Drugood: Out-of-distribution (ood) dataset curator and benchmark for ai-aided drug discovery–a focus on affinity prediction problems with noise annotations. *arXiv preprint arXiv:2201.09637*, 2022.

[17] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang. Graph structure learning for robust graph neural networks, 2020.

[18] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[19] B. Knyazev, G. W. Taylor, and M. R. Amer. Understanding attention and generalization in graph neural networks, 2019.

[20] D. Krueger, E. Caballero, J.-H. Jacobsen, A. Zhang, J. Binas, D. Zhang, R. Le Priol, and A. Courville. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pages 5815–5826. PMLR, 2021.

[21] J. Lee, I. Lee, and J. Kang. Self-attention graph pooling, 2019.

[22] H. Li, X. Wang, Z. Zhang, and W. Zhu. Ood-gnn: Out-of-distribution generalized graph neural network, 2021.

[23] J. Li, D. Cai, and X. He. Learning graph-level representation for drug discovery. *arXiv preprint arXiv:1709.03741*, 2017.

[24] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[25] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

[26] N. Liu, X. Huang, and X. Hu. Accelerated local anomaly detection via resolving attributed networks. In *IJCAI*, pages 2337–2343, 2017.

[27] W. Liu, X. Wang, J. Owens, and Y. Li. Energy-based out-of-distribution detection. *Advances in Neural Information Processing Systems*, 33:21464–21475, 2020.

[28] J. Ma, W. Tang, J. Zhu, and Q. Mei. A flexible generative framework for graph-based semi-supervised learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[29] R. Ma, G. Pang, L. Chen, and A. van den Hengel. Deep graph-level anomaly detection by glocal knowledge distillation. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. ACM, feb 2022.

[30] H. Masnadi-Shirazi, V. Mahadevan, and N. Vasconcelos. On the design of robust classifiers for computer vision. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 779–786. IEEE, 2010.

[31] G. J. McLachlan and T. Krishnan. *The EM algorithm and extensions*, volume 382. John Wiley & Sons, 2007.

[32] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. *CoRR*, abs/1810.02244, 2018.

[33] B. Müller, J. Reinhardt, and M. T. Strickland. *Neural networks: an introduction*. Springer Science & Business Media, 1995.

[34] M. Neumann, R. Garnett, C. Bauckhage, and K. Kersting. Propagation kernels: efficient graph kernels from propagated information. *Machine Learning*, 102(2):209–245, 2016.

[35] J. Nixon, M. W. Dusenberry, L. Zhang, G. Jerfel, and D. Tran. Measuring calibration in deep learning. In *CVPR Workshops*, volume 2, 2019.

[36] Z. Peng, M. Luo, J. Li, L. Xue, and Q. Zheng. A deep multi-view framework for anomaly detection on attributed networks. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[37] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

[38] M. Ren, W. Zeng, B. Yang, and R. Urtasun. Learning to reweight examples for robust deep learning. In *International conference on machine learning*, pages 4334–4343. PMLR, 2018.

[39] Y. Rong, W. Huang, T. Xu, and J. Huang. Dropedge: Towards deep graph convolutional networks on node classification, 2019.

[40] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

[41] Y. Sun, S. Wang, X. Tang, T.-Y. Hsieh, and V. Honavar. Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In *Proceedings of the Web Conference 2020*, pages 673–683, 2020.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[43] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks, 2017.

[44] H. Wang, C. Zhou, X. Chen, J. Wu, S. Pan, and J. Wang. Graph stochastic neural networks for semi-supervised learning. *Advances in Neural Information Processing Systems*, 33:19839–19848, 2020.

[45] Q. Wu, H. Zhang, J. Yan, and D. Wipf. Handling distribution shifts on graphs: An invariance perspective, 2022.

[46] Y.-X. Wu, X. Wang, A. Zhang, X. He, and T.-S. Chua. Discovering invariant rationales for graph neural networks. *arXiv preprint arXiv:2201.12872*, 2022.

[47] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? *CoRR*, abs/1810.00826, 2018.

[48] P. Yanardag and S. Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1365–1374, 2015.

[49] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[50] C. Zhang, J. Bütepage, H. Kjellström, and S. Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.

[51] M. Zhang, Z. Cui, M. Neumann, and Y. Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[52] L. Zhao and L. Akoglu. On using classification datasets to evaluate graph-level outlier detection: Peculiar observations and new insights, 2020.

[53] D. Zügner, A. Akbarnejad, and S. Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &amp Data Mining*. ACM, jul 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Appendix G.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix G.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]

(b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A.

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Appendix E for dataset information. See the supplementary for the codes. They will be released public upon publication.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix D and Appendix E.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See experiment section.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix D.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Appendix E.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Proof for Technical Results

A.1 Derivation of the Learning Objective

Since Eq. 7 is just the unfolding form of Eq. 6, we will give the derivation of Eq. 6 in this section. Consider the KL divergence between $q_\phi(\mathbf{e}|A,X,y)$ and $p_\theta(\mathbf{e}|A,X,y)$:

$$\begin{aligned}
&D_{\mathrm{KL}}[q_\phi(\mathbf{e}|A,X,y)||p_\theta(\mathbf{e}|A,X,y)]\\
=&\int_{\mathbf{e}} q_\phi(\mathbf{e}|A,X,y)\log\frac{q_\phi(\mathbf{e}|A,X,y)}{p_\theta(\mathbf{e}|A,X,y)}d\mathbf{e}\\
=&\mathbb{E}_{q_\phi(\mathbf{e}|A,X,y)}[\log q_\phi(\mathbf{e}|A,X,y)]-\int_{\mathbf{e}} q_\phi(\mathbf{e}|A,X,y)\log\frac{p_\theta(\mathbf{e},A,X,y)}{p_\theta(A,X,y)}d\mathbf{e}\\
=&\mathbb{E}_{q_\phi(\mathbf{e}|A,X,y)}[\log q_\phi(\mathbf{e}|A,X,y)]-\mathbb{E}_{q_\phi(\mathbf{e}|A,X,y)}[\log p_\theta(\mathbf{e},A,X,y)]+\log p_\theta(A,X,y)\\
=&-\mathbb{E}_{q_\phi(\mathbf{e}|A,X,y)}[\log p_\theta(A,X,y|\mathbf{e})]+D_{\mathrm{KL}}[q_\phi(\mathbf{e}|A,X,y)||p(\mathbf{e})]+\log p_\theta(A,X,y)\\
=&-\mathbb{E}_{q_\phi(\mathbf{e}|A,X,y)}[\log p_\theta(A,y|X,\mathbf{e})]+D_{\mathrm{KL}}[q_\phi(\mathbf{e}|A,X,y)||p(\mathbf{e})]+\log p_\theta(A,y|X)\\
=&-\mathbb{E}_{q_\phi(\mathbf{e}|A,X,y)}[\log(p_\theta(A|X,\mathbf{e})p_\theta(y|X,A,\mathbf{e}))]+D_{\mathrm{KL}}[q_\phi(\mathbf{e}|A,X,y)||p(\mathbf{e})]+\log p_\theta(A,y|X).
\end{aligned}\tag{16}$$

Simply move the terms and we can yield:

$$\begin{aligned}
&\log p_\theta(A,y|X)-D_{\mathrm{KL}}[q_\phi(\mathbf{e}|A,X,y)||p_\theta(\mathbf{e}|A,X,y)]\\
=&\mathbb{E}_{q_\phi(\mathbf{e}|A,X,y)}[\log(p_\theta(A|X,\mathbf{e})p_\theta(y|X,A,\mathbf{e}))]\\
&-D_{\mathrm{KL}}[q_\phi(\mathbf{e}|A,X,y)||p(\mathbf{e})]=\mathcal{L}_{\mathrm{ELBO}},
\end{aligned}\tag{17}$$

which is consistent with Eq. 6.

A.2 Proof for Proposition 1

We prove the two sub-propositions separately.

Proof for 1). For one Bernoulli distribution $p(\mathbf{x})=\mathrm{Bernoulli}(\mathbf{x}|\alpha)$, we have $\mathbb{E}_{p(\mathbf{x})}[\mathbf{x}]=\alpha$. Thus, when $q_\phi(\mathbf{e}|\mathbf{A},\mathbf{X},\mathbf{y})$ is instantiated as a Bernoulli distribution, we have:

$$\begin{aligned}
-\mathcal{L}_{cls}&=\sum_{i=1}^{N^{tr}}\mathbb{E}_{q_\phi(\mathbf{e}_i|A_i,X_i,y_i)}[\mathbf{e}_i\log p_\theta(y_i|X_i,A_i)+(1-\mathbf{e}_i)\log p_0(y_i|X_i,A_i)]\\
&=\sum_{i=1}^{N^{tr}}q_\phi(\mathbf{e}_i=1|A_i,X_i,y_i)\log p_\theta(y_i|X_i,A_i)+(1-q_\phi(\mathbf{e}_i=1|A_i,X_i,y_i))\log p_0(y_i|X_i,A_i).
\end{aligned}\tag{18}$$

We can tell that $\mathcal{L}_{cls}$ is just in the re-weighted form, with $q_\phi(\mathbf{e}_i=1|A_i,X_i,y_i)$ acting as a weight for the i-th sample. The deduction is also suitable for $\mathcal{L}_{reg}$.

Proof for 2). Suppose we know the ground truth environments e_i for each training sample, as we are given the ideal recognition model, Eq. 18 can be further written as:

$$\begin{aligned}
-\mathcal{L}_{cls}&=\sum_{i=1}^{N^{tr}}q_\phi(\mathbf{e}_i=1|A_i,X_i,y_i)\log p_\theta(y_i|X_i,A_i)+(1-q_\phi(\mathbf{e}_i=1|A_i,X_i,y_i))\log p_0(y_i|X_i,A_i)\\
&\xlongequal{q^*_\phi(\mathbf{e}|\mathbf{A},\mathbf{X},\mathbf{y})}\sum_{i=1,e_i=1}^{N^{tr}}\log p_\theta(y_i|X_i,A_i)+\sum_{i=1,e_i=0}^{N^{tr}}\log p_0(y_i|X_i,A_i).
\end{aligned}\tag{19}$$

For the optimization objective, we can ignore the second term as $p_0(\mathbf{y}|\mathbf{X},\mathbf{A})$ is a fixed distribution, thus we have:

$$-\mathcal{L}'_{cls}=\sum_{i=1,e_i=1}^{N^{tr}}\log p_\theta(y_i|X_i,A_i)=\sum_{i=1,e_i=1}^{N^{tr}}L(y_i,f_\theta(G_i)),\tag{20}$$

where $G_i = (A_i, X_i)$. As we can see, optimizing over Eq. 20 is just to minimize the empirical loss $\mathbb{E}_{(G,y)\sim p(\mathbf{G},\mathbf{y}|\mathbf{e}=1)}[L(y, f_\theta(G))]$, which is consistent with the debiased learning objective in Eq. 2. As the objective gives full weight to ID data while isolating the outliers (which bring wrong gradients), the classification model can learn to best fit the ID data. The proof for the structure estimation model is just the same.

A.3 Proof for Proposition 2

We prove the two sub-propositions separately.

Proof for 1). Using $\mathcal{L}_{cls}$ as an example, following Eq. 18, we can calculate its gradient on the i-th sample:

$$\frac{\partial - \mathcal{L}_{cls}}{\partial q_\phi(\mathbf{e}_i = 1|A_i, X_i, y_i)} = \log p_\theta(y_i|X_i, A_i) - \log p_0(y_i|X_i, A_i). \tag{21}$$

We optimize on ϕ to minimize $\mathcal{L}_{cls}$, i.e. maximize $-\mathcal{L}_{cls}$. As we have assumed that the classification model $p_\theta(\mathbf{y}|\mathbf{A}, \mathbf{X})$ will assign higher loss to outliers, we have:

$$-\log p_\theta(y_i|X_i, A_i) < -\log p_\theta(y_j|X_j, A_j), \text{ where } e_i = 1, e_j = 0. \tag{22}$$

so we can yield that:

$$\frac{\partial - \mathcal{L}_{cls}}{\partial q_\phi(\mathbf{e}_i = 1|A_i, X_i, y_i)} > \frac{\partial - \mathcal{L}_{cls}}{\partial q_\phi(\mathbf{e}_j = 1|A_j, X_j, y_j)}, \text{ where } e_i = 1, e_j = 0. \tag{23}$$

To maximize $-\mathcal{L}_{cls}$, in this sense, the recognition model will generally learn to assign higher weight for ID data ($e_i = 1$) than the outliers ($e_j = 0$). The proof for the structure estimation model is just the same.

Proof for 2). Suppose we are given the optimal generative models $p^*_\theta(\mathbf{y}|\mathbf{A}, \mathbf{X})$ and $p^*_\theta(\mathbf{A}|\mathbf{X})$ that best fit ID data while performing randomly on the outliers, using $p^*_\theta(\mathbf{y}|\mathbf{A}, \mathbf{X})$ as an example, we have:

$$\log p_\theta(y_j|X_j, A_j) = \log K^{-1}, \text{ where } e_j = 0. \tag{24}$$

where K is the number of target classes. As we have instantiated in Sec. 3.2 that $p_0(\mathbf{y}|\mathbf{A}, \mathbf{X}) = K^{-1}$, we have:

$$\frac{\partial - \mathcal{L}_{cls}}{\partial q_\phi(\mathbf{e}_j = 1|A_j, X_j, y_j)} = 0, \text{ where } e_j = 0. \tag{25}$$

so the recognition model can assign probability 0 to all outliers as they don't impact the return loss. In comparison, for the ID data, we have:

$$\frac{\partial - \mathcal{L}_{cls}}{\partial q_\phi(\mathbf{e}_i = 1|A_i, X_i, y_i)} = \log p_\theta(y_i|X_i, A_i) - \log p_0(y_i|X_i, A_i) > 0, \text{ where } e_i = 1, \tag{26}$$

the inequality is induced based on that the ideal classification model must predict better than random. Thus, the objective is minimized when the recognition model predicts with probability 1 for all ID data. In conclusion, there exists a recognition model that achieves the minimal objective while ideally predict the environment variable.

B More Related Works

Graph neural networks. GNNs [18, 43] are increasingly attracting attention in recent years because of their notable success in graph representation learning [12]. They generally utilize a message-passing paradigm, which combines node features and graph structure to update node embeddings. A series of GNNs have been proposed to achieve state-of-the-art performance on various graph tasks, including node classification [18], link prediction [24], and graph classification [51, 10]. Despite their great success, existing works usually pose the i.i.d. assumption across training and testing data. Therefore, GNNs' performance may degenerate by a large extend when the training dataset is mixed with outliers drawn from other distributions [3]. Besides, incorrect predictions can also be made because of distribution shifts on testing dataset. Thus, it's critical to develop methods to boost GNNs' robustness to outliers as well as detect OOD samples on testing dataset.

Trustworthy GNNs. With widespread utilization of GNNs, their trustwothiness has raised public concern these days. Previous works have shown that GNNs are vulnerable to simple perturbations [53,

41] on graph structures and node attributes. Therefore, a series of works are proposed to promote GNNs' robustness to adversarial attacks through methods like data augmentation [11] and structure learning [17, 8]. Besides, the OOD generalization capability of GNNs, either at node-level [45] or graph-level [22, 46], are increasingly drawing attention recently. We emphasize that our work is orthogonal to them for that: first, we are not focusing on the adversarial robustness, but robustness to outliers, which are drawn from other distributions and should be filtered out during training; second, we are trying to detect and reject OOD samples rather than promote GNNs' generalization capability.

C GraphDE Training Procedure

Using GraphDE-a as an example, its training procedure is summarized in Alg. 1.

Algorithm 1: The training procedure of GraphDE-a.

Input: Dataset $\mathcal{D}^{tr} = \{(A_i, X_i, y_i)\}_{i=1}^{N^{tr}}$, number of training epochs E, batch size B.

Initialize parameters θ;
for epoch in $1, 2...E$ **do**
  Sample data batches $\mathcal{B} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_k\}$ with batch size B from $\mathcal{D}^{tr}$;
  for $i \leftarrow 1...k$ **do**
    for (A_j, X_j, y_j) in $\mathcal{D}_i$ **do**
      Compute the posterior $p_\theta(e_i|A_i, X_i, y_i)$ by Eq. 10;# E-step.
      Compute $p_\theta(y_i|A_i, X_i) = \text{GNN}(A_i, X_i)$;
      Sample one disconnected node pairs for each existing edge;# Negative sampling.
      Compute $p_\theta(A_i|X_i)$ by Eq. 11 or Eq. 12; # Compute edge probabilities using LSM or CosLSM.
    Compute the total batch loss $\mathcal{L}$ by Eq. 7;
    Backpropagate $\mathcal{L}$ and optimize parameters θ;# M-step.

Output: Trained parameters θ.

D Implementation Details

We present our implementation details here for reproducibility. Our model and all the baselines are implemented with Python 3.8, PyTorch 1.10.0 and PyTorch Geometric 2.1.0. All experiments are run on a Tesla V100.

We choose GCN [18], GAT [43], TopKPool [19] and SAGPool [21] as our backbone models, whose implementations are provided by the original papers. For robust GNN baselines DropEdge [39] and GRAND [11], and the deep learning based OOD detection baselines OCGIN [52] and GLocalKD [29], we also refer to their official implementations. Specifically, we use GAT as the backbone for our model and baselines, except on SPMotif, which we have tried all the backbones.

D.1 Details for Debiasing Experiments

Classification model architectures. For debiased learning experiments, the baseline models GCN, GAT, TopKPool, SAGPool are implemented in the following manner:

- For GCN and GAT, we use two GNN layers with hidden size 64 by default. TopKPool and SAGPool are comprised of two blocks, each of which consists of one GIN layer and one pooling layer.
- For DrugOOD and MNIST, we use the global mean pool to extract the graph-level representation, while on Collab and SPMotif, the global max pool is adopted.
- We use two MLP layers with hidden size 64 after the global pooling layer for graph classification.
- The activation function is ReLU.

Structure estimation model architectures.

- For SPMotif and Collab, we use LSM as the structure estimation model. The transformation matrix $\boldsymbol{U}$'s output size is 64.

- For MNIST-75sp and DrugOOD, we utilize CosLSM as the structure estimation model. The transformation matrix $\boldsymbol{U}_i$'s output size is 64. We use 4 heads by default.
- The activation function is ReLU.

Training Details. We use mini-batch gradient descent to optimize our model and the baselines. More concretely, we use the cross entropy loss (or negative log likelihood) and the Adam optimizer is adopted for optimization. The total training epoch number is fixed at 400. We use early stopping of 20 epochs.

Testing Details. We report the ID test accuracy achieved by the epoch that gives the highest accuracy on validation dataset.

Hyperparameters. We adopt grid search to tune the hyperparameters. The learning rate is searched within $\{0.1, 0.01, 0.001, 0.0001\}$, dropout is searched within $\{0.0, 0.1, 0.5\}$. Specifically, the hyperparameters are set as follows.

- For GCN and GAT, the learning rate is 0.01 and the batch size is 64. No dropout is used. For dataset MNIST and DrugOOD, batchnorm is used.
- For SAGPool and TopKPool, the min score is set to 0.001, i.e. nodes with scores higher than 0.001 will be selected for propagation.
- The dropedge rate is chosen within $\{0.1, 0.2, 0.3, 0.4, 0.5\}$.
- For GRAND, the propagation order is chosen within $\{1, 2, 3, 4, 5\}$, while the dropnode rate is chosen within $\{0.05, 0.1, 0.3, 0.5\}$.
- For our model, the hyperparameters are set as default.

D.2 Details for Detection Experiments

Architectures. For detection experiments, our model GraphDE is implemented in the same manner as in the debiasing setting.

Training Details. For MSP and our model GraphDE, the OOD detector is extracted from the trained model, i.e. the training paradigm is the same as in the debiasing setting. The two-stage graph kernel baselines don't require training. For OCGIN, we adopt its modified one-class objective and train for 100 epochs with Adam optimizer. For GLocalKD, we adopt the same setting as in the original paper [29] and train for 50 epochs with Adam optimizer.

Testing Details We feed equal amounts of ID and OOD data into the trained model and calculate the three metrics: AUROC, AUPR, and FPR95 to evaluate the OOD detection performance of our model. Area under the receiver operating characteristic curve(AUROC) can be considered as the probability that an outlier is assigned a higher OOD score than an ID sample, so a higher AUROC score is better. Area under the precision-recall curve(AUPR) is a useful performance metric for imbalanced data, which takes into account the base rate of outliers. FPR95 is the false positive rate of OOD examples when the true positive rate for ID examples is 95%, which represents performance at one strict threshold.

Hyperparameters. The biased ratio is set as 30% for fair comparison. For the two-stage graph kernel baselines, the kernel is chosen within {PK [34], WL [40]}, while the detector is chosen within {OCSVM [6], LOF [5]}.

- For OCGIN, we adopt 3 GIN [47] layers with hidden size 64. The learning rate is set as 0.001. We do not use weight decay or dropout. Batch normalization is turned on or off manually for better detection performance.
- For GLocalKD, we adopt 2 GraphConv [32] layers with hidden size 512 and output size 256. The learning rate is set as 0.0001. We adopt batch normalization and the dropout is set as 0.3 during training. We do not use weight decay.
- For graph kernels, the WL iteration number is set as 5, the bin width of PK is set as 0.1.
- For LOF, the leaf number and the number of neighbors is set as 30 and 20.
- For our model, the hyperparameters are set as default.

Table 3: Statistics of experiment datasets. The OOD types of the datasets have been discussed in the main text. # Class is the target class number. The latter 5 column captions denote the numbers of training graphs, validation graphs, ID testing graphs, OOD mixed graphs, and OOD testing graphs we use in the experiments, respectively.

Dataset	OOD Type	# Class	# Train	# Val	# ID Test	# OOD Mix	# OOD Test
SPmotif	Spurious Connection	3	9000	3000	6000	9000	6000
MNIST-75sp	Gaussian Noise	10	3000	1000	2000	3000	2000
Collab	Graph Size	3	1000	300	500	501	500
DrugOOD	Scaffold	2	1000	500	500	1000	500

E Dataset Information

We present detailed information for our used datasets concerning the data collection, preprocessing, and statistic information. An overview of the datasets is represented in Tab. 3.

E.1 Dataset Information

Synthetic dataset. SPMotif is a synthetic dataset in which each graph is composed of one base (denoted by $S = 0, 1, 2$) and one motif (denoted by $C = 0, 1, 2$). The graph label y is solely determined by the motif C. We manually construct spurious correlations between the base S and label y to simulate distribution shifts. Concretely, we sample each motif from a uniform distribution, while the distribution of its base is determined by:

$$P(S) = \begin{cases} b, & \text{if } S = C, \\ (1-b)/2, & \text{otherwise.} \end{cases} \tag{27}$$

We change the mapping relationship to create different spurious correlations to simulate distribution shift, namely, the distribution of S in OOD testing dataset is:

$$P(S) = \begin{cases} b, & \text{if } S \equiv C + 1 (mod\, 3), \\ (1-b)/2, & \text{otherwise.} \end{cases} \tag{28}$$

To avoid data leakage as discussed in the main text, the distribution of S in OOD mixed dataset is:

$$P(S) = \begin{cases} b, & \text{if } S \equiv C + 2 (mod\, 3), \\ (1-b)/2, & \text{otherwise.} \end{cases} \tag{29}$$

which is different to both ID and OOD testing datasets. The ratio b is set as 0.9 for ID, OOD testing, and OOD mixed datasets. We employ 9000 training graphs, 9000 OOD mixed graphs, 3000 validation graphs, and 6000 ID and OOD testing graphs. For debiased learning which only needs OOD mixed dataset, we directly use Eq. 28 to generate outliers instead.

Real-world datasets. We adopt three real-world datasets MNIST-75sp, Collab and DrugOOD to evaluate our model.

- MNIST-75sp is converted from the original dataset MNIST, each image of which depicts a handwritten digit from 0 to 9. The converted super-pixel graphs have at most 75 nodes, with node features set as super-pixel coordinates and intensity, while edges are the spatial distance between the nodes. We add Gaussian noise $\mathcal{N}(0, 0.3)$ to node features for outliers in the training dataset, while using Gaussian noise $\mathcal{N}(0, 0.6)$ to simulate distribution shifts in the OOD testing dataset. The number of graphs used in the training and validation datasets are 3000 and 1000, and we use 2000 ID and OOD testing datasets. For debiased learning which only needs OOD mixed dataset, we add Gaussian noise $\mathcal{N}(0, 0.45)$ to node features.
- Collab is a scientific-collaboration dataset consisting of 5000 researchers' ego network. Each node represents a researcher, and the edge represents the collaboration. We use the graph size to simulate distribution shift, with graphs of [45, 80] nodes denoted as ID, graphs of (80, 100] nodes treated as OOD mixed data, while those of more than 100 nodes treated as OOD testing data. Specifically, We use 1000 samples for training, 300 samples for validation, 501 samples for training outliers, and 500 samples to formulate ID and OOD testing dataset. For debiased learning, in order to create larger distribution shifts, we treat graphs with less than 45 nodes as ID data, and those with more than 90 nodes as OOD mixed data (i.e. training outliers).

- DrugOOD is a systematic OOD dataset curator and benchmark for drug discovery, providing large-scale, realistic, and diverse datasets for graph OOD learning problems. The OOD property lies in the scaffold difference. Specifically, we utilize the provided built-in configuration file, namely, *lbap_general_ic50_scaffold.py* to generate the dataset. As described in their paper [16], DurgOOD generates domain descriptor (the molecular size is applied to the domain of scaffold) for each domain, and then sort the domains with descriptors. Then the sorted domains are sequentially divided into training set, OOD validation and testing set (with a proportion of 6:2:2). We use these three sets to construct our experimental datasets since they have different data distributions. Specifically, we randomly select 1000, 300, 500 graphs from the original training set to formulate the ID training, validation and testing datasets. Our OOD mixed dataset is consisted of 1000 graphs from the original OOD validation dataset. Finally, 500 graphs are drawn from the original OOD testing dataset to compose our OOD testing dataset. For debiased learning, we treat the original training dataset as ID data, and the original testing dataset as training outliers, to create larger distribution shifts.

E.2 Dataset Preprocessing

All the datasets we use in the experiment are directly collected from the source except DrugOOD, which is generated by ourselves with the help of the provided dataset curator. Besides, the node features of SPMotif are set as 4-dimensional uniform random features. As Collab is originally comprised of plain graphs without node features, we use one-hot node degrees as features for them.

F More Experiment Results

F.1 Detection Results

Table 4: Detection results on SPMotif. The biased ratio is fixed as 30% for a fair comparison. We report the mean and standard deviation for all the detectors, except the two-stage models, which are invariant to different random seeds. We use GraphDE-a as the OOD detector.

Detector	AUROC↑	AUPR↑	FPR95↓
MSP	50.06±0.38	50.19±0.27	95.35±0.26
WL+OCSVM	50.92	48.74	94.27
WL+LOF	50.44	49.51	94.75
PK+OCSVM	47.19	45.73	93.87
PK+LOF	51.15	49.20	93.80
OCGIN	47.98±4.33	47.49±0.70	88.20±3.81
GLocalKD	52.40±0.21	51.54±0.08	90.60±0.38
ours	52.56±0.38	51.69±0.27	91.33±0.25

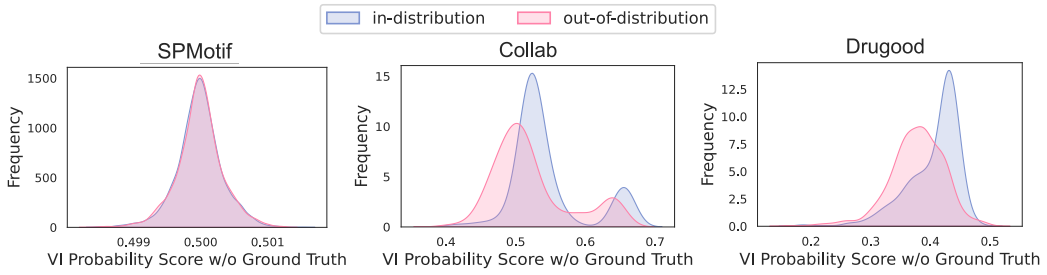
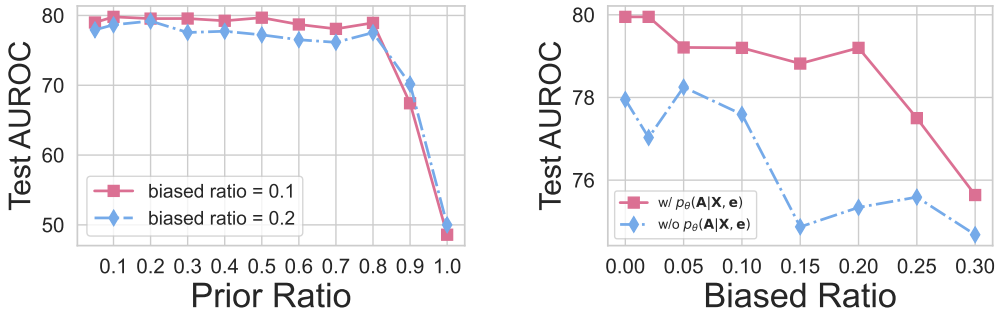


Figure 6: Testing OOD score distribution. We use the GraphDE-a variant and the biased ratio is set as 20% for all the experiments.



(a) Prior Ratio

(b) Structure Estimation Model

Figure 7: Further studies (on DrugOOD). We use the GraphDE-a variant for all the experiments.

Table 5: Space/Time complexity evaluation results on SPMotif and DrugOOD. The default hyperparameter setting is adopted during evaluation. "Running Time" denotes the training time per epoch.

Algorithm	**SPMotif**		**DrugOOD**	
	Running Time (s)	GPU Usage (MB)	Running Time (s)	GPU Usage (MB)
Backbone	11.896	2,013	0.211	1,781
DropEdge	14.793	2,013	0.570	1,711
GRAND	33.630	4,319	6.034	3,039
GraphDE-v	40.622	3,961	2.096	9,623
GraphDE-a	36.011	3,961	2.687	9,351

Tab. 4 reports GraphDE's detection performance on the synthetic dataset SPMotif. As is presented in the table, all the models, including the max softmax score(MSP), two-stage kernel methods, OCGIN, GLocalKD, and GraphDE, have a poor performance. Despite the fact that GraphDE performs the best on 2 out of 3 benchmarks, it only provides a slight improvement, which is due to the properties of the dataset. More concretely, this dataset is unsuitable for OOD detection since the distribution shift is dominated by the label information, opposed to that the OOD detectors focus on detecting input graphs with abnormal structures or node features.

Fig.6 demonstrates the detection performance of GraphDE-a. As is shown in the figure, the ID data is assigned with a higher score on Collab and DrugOOD, while OOD data tends to get a relatively low score. However, the OOD score distributions of ID and OOD data are almost the same on SPMotif, which is consistent with the results in Tab. 4.

F.2 Further Study

We supplement extra experimental results on DrugOOD to support GraphDE's effectiveness.

Impact of prior ratio. Like that in Sec. 4.3, we tune and study the impact of prior ratio $p(\mathbf{e})$ in the KL loss $\mathcal{L}_{kl}$. As can be seen from Fig 7(a) , the debiasing performance doesn't fluctuate much from 0.05 to 0.8, while it drops dramatically when the prior ratio goes higher than 0.8. In addition, the test AUROC peaks at 0.1 for biased ratio 0.1, and 0.2 for biased ratio 0.2, exactly when $\mathcal{L}_{kl}$ becomes 0, proving the validity of the KL loss.

Ablation study. To study the effect of the structure regularization loss $\mathcal{L}_{reg}$, we remove $\mathcal{L}_{reg}$ in the objective and plot Fig 7(b). The figure clearly shows the sharp decline in all biased ratios. Especially, the test AUROC undergoes a descent by nearly 5% when the biased ratio is 15%. All of these verify the feasibility of the structure estimation model to help discriminate and down-weight outliers in the training procedure.

F.3 Algorithm Complexity

We provide additional empirical results of space/time usage as shown in Table 5 in this section to gain deeper insights into GraphDE.

Generally, we can conclude that the memory usage of GraphDE on both datasets is reasonable, as they can be accommodated by most of GPUs today. Besides, the table shows that major additional memory footprint of GraphDE comes from the structure estimation model, such that the memory overhead is much more obvious on DrugOOD (consists of graphs with larger number of nodes and edges) than that on SPMotif.

As for the time complexity, using results on DrugOOD as an example, the models usually converge and early stop at around 150 epochs. So it takes at around 5 minutes to train our GraphDE model on DrugOOD, which can achieve a 5% test accuracy improvement over the backbone. Relatively, we believe this is a valuable time-accuracy trade-off. Besides, we find that GraphDE-v is obviously faster than GraphDE-a, this is because it utilizes simple learnable scalars during training and does not need to calculate the posterior analytically. So GraphDE-v can be a good choice if we need to train a GraphDE model in limited time. In comparison to other two plug-in modules, we find that GraphDE is much faster than GRAND on DrugOOD, while on par with it on SPMotif. Besides, it is slower than DropEdge but with much better testing performance.

G Limitations and Potential Negative Impacts

Currently, our framework cannot readily incorporate vectorized data like images or texts. Besides, more complex model instantiations may be developed to achieve better debiasing and OOD detection performance on graph data, which we leave for future work.

As we focus on developing trustworthy GNNs, we believe that the negative impacts of our work are small compared to its contributions. However, it can still raise problems like data fairness due to its re-sampling strategy to conduct debiasing. Besides, its robustness as an OOD detector should be studied in-depth as future work, since malicious attackers may fool GraphDE to treat OOD data as ID data, leading to potential performance degradation in practice.