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:

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

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 [17, 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 [17], link prediction [24], and graph classification [10]. Despite their great success, recent evidence shows that common GNNs exhibit limited power when training data is insufficient with missing edges and labels [20]. And, existing works usually pose the i.i.d. assumption across training and testing data, which does not necessarily hold in practice [45]. 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 and Robust 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 [8]. Besides, the OOD generalization capability of GNNs, either at node-level [45] or graph-level [22, 49], 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 [17], GAT [43], TopKPool [18] 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 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 [46] 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

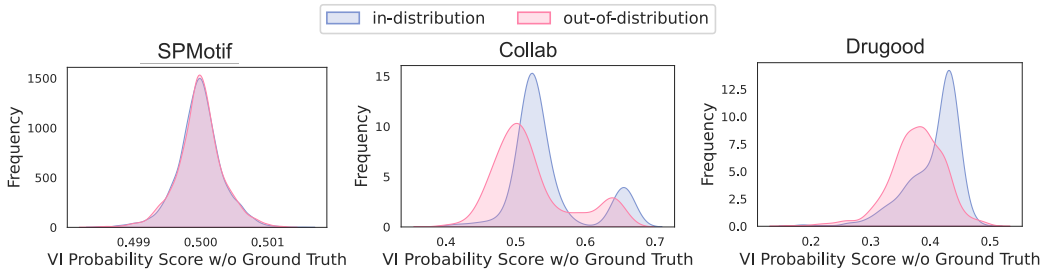
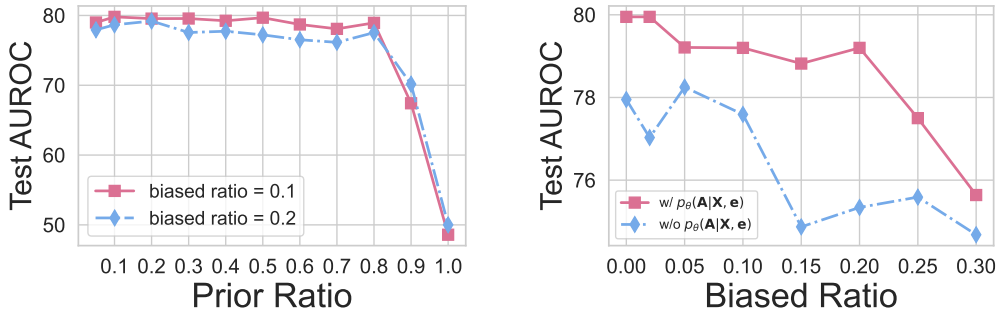


Figure 6: Testing OOD score distribution. We use the GraphDE-a variant and the biased ratio is set as 20% for all the experiments.



(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.