A. Appendix

A.1. Derivation of the Estimator in Theorem 2.1 and Eqn. 6

$$
\begin{aligned}
I(\mathbf{Z}_c \to y) =& \int_{\mathbf{Z}_c} P(\mathbf{Z}_c)\Big(\sum_y P(y|do(\mathbf{Z}_c)) \log P(y|do(\mathbf{Z}_c))\Big) d\mathbf{Z}_c \\
& - \sum_y \int_{\mathbf{Z}_c} P(\mathbf{Z}_c) P(y|do(\mathbf{Z}_c)) d\mathbf{Z}_c \cdot \\
& \log \Big(\int_{\mathbf{Z}_c} P(\mathbf{Z}_c) P(y|do(\mathbf{Z}_c)) d\mathbf{Z}_c \Big) \\
=& \frac{1}{N_c} \sum_{i=1}^{N_c} \sum_y \Big(\frac{1}{N_x N_s N_z} \sum_{k=1}^{N_x} \sum_{j=1}^{N_s} \sum_{n=1}^{N_z} P(y|\mathbf{A}^{(ikjn)}, \mathbf{X}^{(k)}) \Big) \cdot \\
& \log \Big(\frac{1}{N_x N_s N_z} \sum_{k=1}^{N_x} \sum_{j=1}^{N_s} \sum_{n=1}^{N_z} P(y|\mathbf{A}^{(ikjn)}, \mathbf{X}^{(k)}) \Big) \\
& - \sum_y \Big(\frac{1}{N_c N_x N_s N_z} \sum_{i=1}^{N_c} \sum_{k=1}^{N_x} \sum_{j=1}^{N_s} \sum_{n=1}^{N_z} P(y|\mathbf{A}^{(ikjn)}, \mathbf{X}^{(k)}) \Big) \cdot \\
& \log \Big(\frac{1}{N_c N_x N_s N_z} \sum_{i=1}^{N_c} \sum_{k=1}^{N_x} \sum_{j=1}^{N_s} \sum_{n=1}^{N_z} P(y|\mathbf{A}^{(ikjn)}, \mathbf{X}^{(k)}) \Big) \\
=& \frac{1}{N_c N_x N_s N_z} \Big[\sum_{i=1}^{N_c} \sum_y \Big(\sum_{k=1}^{N_x} \sum_{j=1}^{N_s} \sum_{n=1}^{N_z} P(y|\mathbf{A}^{(ikjn)}, \mathbf{X}^{(k)}) \Big) \cdot \\
& \log \Big(\frac{1}{N_x N_s N_z} \sum_{k=1}^{N_x} \sum_{j=1}^{N_s} \sum_{n=1}^{N_z} P(y|\mathbf{A}^{(ikjn)}, \mathbf{X}^{(k)}) \Big) \\
& - \sum_y \Big(\sum_{i=1}^{N_c} \sum_{k=1}^{N_x} \sum_{j=1}^{N_s} \sum_{n=1}^{N_z} P(y|\mathbf{A}^{(ikjn)}, \mathbf{X}^{(k)}) \Big) \cdot \\
& \log \Big(\frac{1}{N_c N_x N_s N_z} \sum_{i=1}^{N_c} \sum_{k=1}^{N_x} \sum_{j=1}^{N_s} \sum_{n=1}^{N_z} P(y|\mathbf{A}^{(ikjn)}, \mathbf{X}^{(k)}) \Big) \Big]
\end{aligned}
$$

A.2. Further Implementation Details

Datasets. BA-shapes was created with a base Barabasi-Albert (BA) graph containing 300 nodes and 80 five-node "house"-structured network motifs. Tree-cycles were built with a base 8-level balanced binary tree and 80 six-node cycle motifs. Mutag [2] and NCI1 [26] are for graph classification tasks. Specifically, Mutag contains 4337 molecule graphs, where nodes represent atoms, and edges denote chemical bonds. It contains the non-mutagenic and mutagenic class, indicating the mutagenic effects on Gram-negative bacterium Salmonella typhimurium. NCI1 consists of 4110 instances; each chemical compound screened for activity against non-small cell lung cancer or ovarian cancer cell lines. The statistics of four datasets are presented in Table 5. Note that, we report the average number of nodes and the average number of edges over all the graphs for the real-world datasets.

Model architectures. For classification architectures, we use the same setting as prior works [11, 32]. Specifically, for node classification, we apply three layers of GCNs with output dimensions equal to 20 and perform concatenation to the output of three layers, followed by a linear transformation to obtain the node label. For graph classification, we employ three layers of GCNs with dimensions of 20 and perform global max-pooling to obtain the graph representations. Then a linear transformation layer is applied to obtain the graph label. Figure 6 (a) and 6 (b) are the model architectures for node classification and graph classification, receptively.

Figure 6 (c) depicts the model architecture of *OphicX* for generating explanations. For the inference network, we applied a three-layer GCN with output dimensions 32, 32, and 16. The generative model is equipped with a two-layer MLP and an inner product decoder. We trained the explainers using the Adam optimizer [7] with a learning rate of 0.003 for 300epochs. Table 7 shows the detailed data splitting for model training, testing, and validation. Note that both classification models and our explanation models use the same data splitting. See Table 8 for our hyperparameter search space. Table 6 reports the model accuracy on four datasets, which indicates that the models to be explained are performed reasonably well. Unless otherwise stated, all models, including GNN classification models and our explanation model, are implemented using PyTorch [9] and trained with Adam optimizer.

Table 5. Data Statistics of Four Datasets.

DATASETS	BA-SHAPES	TREE-CYCLES	MUTAG	NCI1
#GRAPHS	1	1	4,337	4,110
#NODES	700	871	29	30
#EDGES	4,110	1,950	30	32
#LABELS	4	2	2	2

Table 6. Model Accuracy of Four Datasets (%).

DATASETS	BA-SHAPES	TREE-CYCLES	MUTAG	NCI1
ACCURACY	94.1	97.1	88.5	78.6

Table 7. Data Splitting for Four Datasets.

DATASETS	#OF TRAINING	#OF TESTING	#OF VALIDATION
BA-SHAPES	300	50	50
TREE-CYCLES	270	45	45
MUTAG	3,468	434	434
NCI1	3,031	410	411

Negative ELBO term. The negative ELBO term is defined as Eqn. 8:

[9]https://pytorch.org

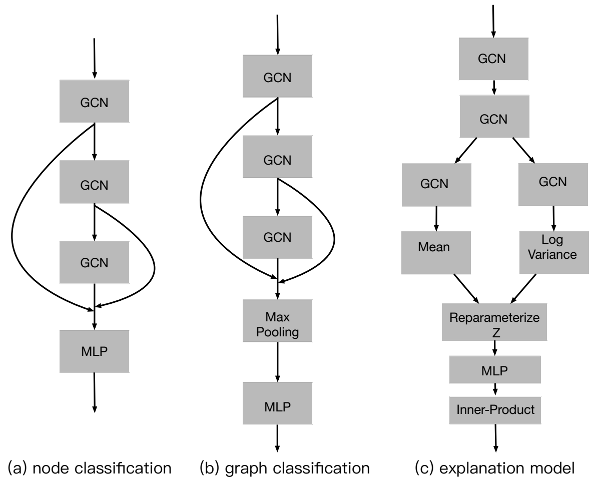


Figure 6. Model architectures.

Table 8. Hyperparameters and Ranges

HYPERPARAMETER	RANGE
CAUSAL DIMENSION $\mathbf{D}_c$	$\{1, 2, 3, \cdots, 8\}$
NEGATIVE ELBO λ_1	$\{0, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1\}$
SPARSITY λ_2	$\{0, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1\}$
FIDELITY λ_3	$\{0, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1\}$

$$\mathcal{L}_{\mathbf{VGAE}} = \mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}[\log p(\mathbf{A}|\mathbf{Z})] - \mathbf{KL}[q(\mathbf{Z}|\mathbf{X},\mathbf{A}) \parallel p(\mathbf{Z})], \quad (8)$$

where $\mathbf{KL}[q(\cdot) \parallel p(\cdot)]$ is the Kullback-Leibler divergence between $q(\cdot)$ and $p(\cdot)$. The Gaussian prior is $p(\mathbf{Z}) = \prod_i p(\mathbf{z}_i) = \prod_i \mathcal{N}(\mathbf{z}_i|0, \mathbf{1})$. We follow the reparameterization trick in [8] for training.

Log-odds difference. We measure the resulting change in the pre-trained GNNs' outcome by computing the difference in log odds and investigate the distributions over the entire test set. The log-odds difference is formulated as:

$$\Delta\text{log-odds} = \text{log-odds}\,(f(G)) - \text{log-odds}\,(f(G_c)) \quad (9)$$

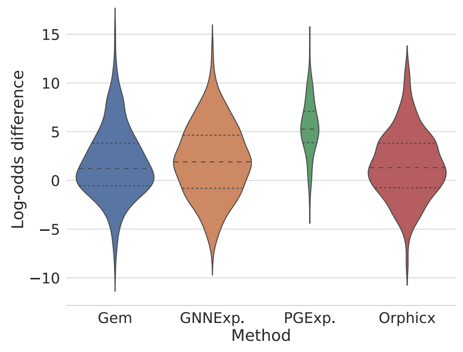
where $\text{log-odds}(p) = \mathbf{log}\left(\frac{p}{1-p}\right)$, and $f(G)$ and $f(G_c)$ are the outputs of the pre-trained GNN. Figure 7 depicts the distributions of log-odds difference over the entire test set for the real-world datasets.

A.3. More Experimental Results

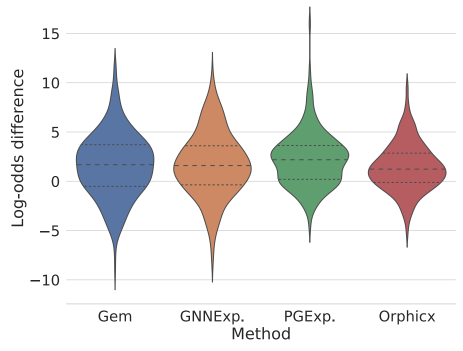
Log-odds difference on the real-world datasets. Figure 7 depicts the distributions of log-odds difference over the entire test set for the real-world datasets. We can observe that the log-odds difference of *OrphicX* is more concentrated around 0, which indicates *OrphicX* can well capture the most relevant subgraphs towards the predictions by the pre-trained GNNs.

Table 9. Causal Evaluation (%).

	Mutag			NCI1		
R (edge ratio)	0.7	0.8	0.9	0.7	0.8	0.9
original	77.2	78.8	83.2	77.1	81.3	85.4
deconfounder	67.1	71.5	81.5	71.6	79.2	87.3



(a) MUTAG



(b) NCI1

Figure 7. Explanation Performance with Log-Odds Difference. *OrphicX* consistently achieves the best performance overall (*denser distribution around* 0 *is better*).

More visualization results. Figure 8 plots the visualized explanations of different methods on BA-shapes. The "house" in green is the ground-truth motif that determines the node labels. The red node is the target node to be explained. By looking at the explanations for a target node (the instance on the left side), shown in Figure 8, *OrphicX* can successfully identify the "house" motif that explains the node label ("middle-node" in red), when $K = 6$, while GNNExplainer wrongly attributes the prediction to a node (in orange) that is out of the "house" motif. For the right one, *OrphicX* consistently performs well, while Gem and GNNExplainer both fail when $K = 6$. Figure 9 plots more visualized explanations of different methods on Mutag.

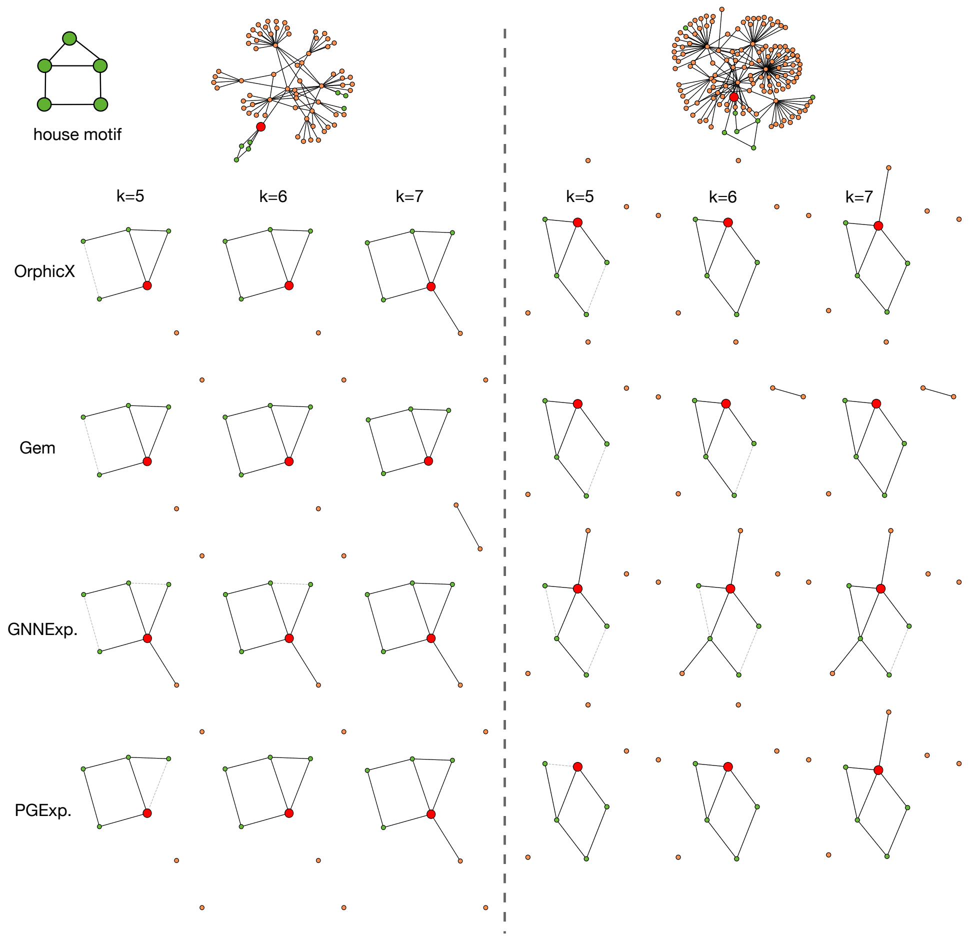


Figure 8. Explanation comparisons on BA-shapes. The "house" in green is the ground-truth motif that determines the node labels. The red node is the target node to be explained (better seen in color).

Causal evaluation. To further verify that the generated explanations are causal and therefore robust to distribution shift in the confounder (i.e., the node attributes $\mathbf{X}$), we construct harder versions of both datasets. Specifically, we use k-means (k=2) to split the dataset into two clusters according to the node attributes. In Mutag, we use the cluster with 3671 graph instances for explainer training and validation; we evaluate the explaining accuracy of the trained explainer on the other cluster with 665 instances. In NCI1, we use the cluster with 3197 graph instances to train an explainer, in which the training set contains 2558 instances and the validation set contains 639 instances; the explaining accuracy is evaluated with the other cluster with 906 instances. See Table 9 for details. We can observe that our approach is indeed robust to the distribution shift in the confounder.

Information flow measurements. To validate Theorem 2.1, we evaluate the information flow of the causal factors ($\mathbf{Z}_c = \mathbf{Z}[1:3]$) and the spurious factors ($\mathbf{Z}_s = \mathbf{Z}[4:16]$) corresponding to the model prediction, respectively. Figure 10a shows that, as desired, the information flow from the causal factors to the model prediction is large while the information flow from the spurious factors to the prediction is small.

Ablation study. We inspect the explanation performance for our framework as an ablation study for the loss function proposed. We empirically prove the need for different forms of regularization leveraged by the *OrphicX* loss function. In particular, we compute the average explanation accuracy of 3 runs. Table 10 shows the explanation accuracy of removing a particular regularization term for Mutag and NCI1, respectively. We observe considerable performance gains from introducing the VGAE ELBO term, sparsity, and fidelity penalty. In summary, these results empirically motivate the need for different forms of regularization leveraged by the *OrphicX* loss function.

Efficiency evaluation. *OrphicX*, Gem, and PGExplainer can explain unseen instances in the inductive setting. We measure the average inference time for these methods. As

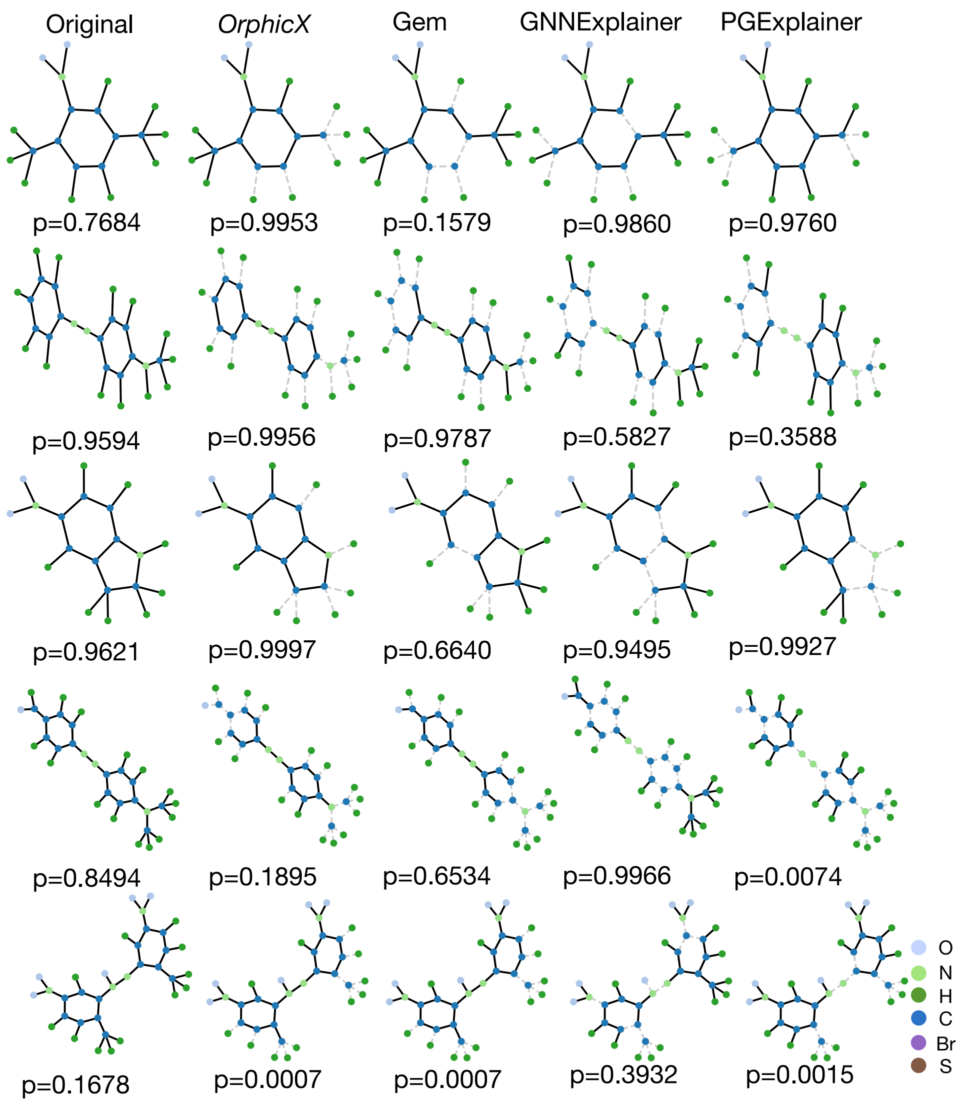


Figure 9. Explanation Visualization (MUTAG): p is the corresponding probability of being classified as Mutagenic class by the pre-trained GNN. The graphs in the first column are the target instances to be explained. The solid edges in other columns are identified as 'important' by corresponding methods. The closer the probability to that of the target instance, the better the explanation is.

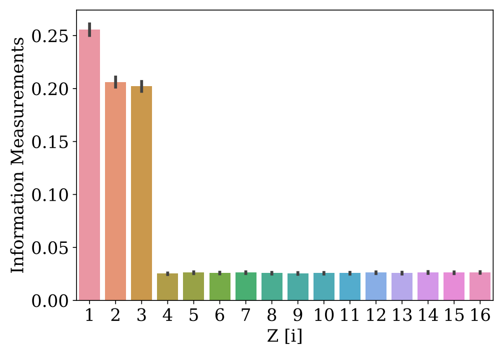
Table 10. Ablation Studies for Different Regularization Terms (%).

TYPE	CAUSAL INFLUENCE	ELBO	SPARSITY	FIDELITY	MUTAG	NCI1
OrphicX	✓	✓	✓	✓	**0.854**	**0.832**
A	✓	✓	✓		0.829	0.633
B	✓	✓		✓	0.804	0.824
C	✓		✓	✓	0.594	0.633

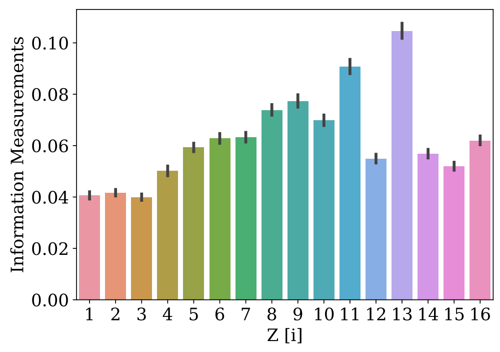
GNNExplainer explains an instance at a time, we measure its average time cost per explanation for comparisons. As reported in Table 11, we can conclude that the learning-based explainers such as *OrphicX*, Gem, and PGExplaienr are more efficient than GNNExplainer. These experiments were performed on an NVIDIA GTX 1080 Ti GPU with an Intel Core i7-8700K processor.

A.4. More related work on GNN interpretation

Several recent works have been proposed to provide explanations for GNNs, in which the most important features (e.g., nodes or edges or subgraphs) of an input graph are selected as the explanation to the model's outcome. In essence, most of these methods are designed for generating input-



(a) MUTAG



(b) MUTAG w/o Causal Term

Figure 10. Information Flow Measurements. Figure 10a reports the information flow measurements in the hidden space, where i denotes the ith dimension. Figure 10b reports the ones while the causal influence term was removed from the loss function.

Table 11. Explanation Time of Different Methods (Per Instance (ms)).

DATASETS	BA-SHAPES	TREE-CYCLES	MUTAG	NCI1
OrphicX	0.61	2.31	0.01	0.02
GEM	0.67	0.50	0.05	0.03
GNNEXPLAINER	260.2	206.5	253.2	262.4
PGEXPLAINER	6.9	6.5	5.5	5.4

dependent explanations. GNNExplainer [32] searches for soft masks for edges and node features to explain the predictions via mask optimization. [21] extended explainability methods designed for CNNs to GNNs. PGM-Explainer [25] adopts a probabilistic graphical model and explores the dependencies of the explained features in the form of conditional probability. SubgraphX explores the subgraphs with Monte Carlo tree search and evaluates the importance of the subgraphs with Shapley values [35]. In general, these methods explain each instance individually and can not generalize to the unseen graphs, thereby lacking a global view of the target model.

A recent study has shown that separate optimization for each instance induces hindsight bias and compromises faithfulness [23]. To this end, PGExplainer [14] was proposed to learn a mask predictor to obtain edge masks for providing instance explanations. XGNN [34] was proposed to investigate graph patterns that lead to a specific class. GraphMask [23] is specifically designed for GNN-based natural language processing tasks, where it learns an edge mask for each internal layer of the learning model. Both these approaches require access to the process by which the target model produces its predictions. As all the edges in the dataset share the same predictor, they might be able to provide a global understanding of the target GNNs. Our work falls into this line of research, as our objective is to learn an explainer that can generate compact subgraph structures contributing to the predictions for any input instances. Different from existing works, we seek faithful explanations from the language of causality [19].