RegExplainer: Generating Explanations for Graph Neural Networks in Regression Tasks

Jiaxing Zhang
New Jersey Institute of Technology
jz48@njit.edu

Zhuomin Chen
Florida International University
zchen051@fiu.edu

Hao Mei
Arizona State University
hmei7@asu.edu

Dongsheng Luo
Florida International University
dluo@fiu.edu

Hua Wei
Arizona State University
hua.wei@asu.edu

Abstract

Graph regression is a fundamental task that has gained significant attention in various graph learning tasks. However, the inference process is often not easily interpretable. Current explanation techniques are limited to understanding GNN behaviors in classification tasks, leaving an explanation gap for graph regression models. In this work, we propose a novel explanation method to interpret the graph regression models (XAIG-R). Our method addresses the distribution shifting problem and continuously ordered decision boundary issues that hinder existing methods away from being applied in regression tasks. We introduce a novel objective based on the information bottleneck theory and a new mix-up framework, which could support various GNNs in a model-agnostic manner. Additionally, we present a contrastive learning strategy to tackle the continuously ordered labels in regression tasks. We evaluate our proposed method on two benchmark datasets and a real-life dataset introduced by us, and extensive experiments demonstrate its effectiveness in interpreting GNN models in regression tasks.

1 Introduction

Graph Neural Networks [1] (GNNs) have become a powerful tool for learning knowledge from graph-structure data and achieved remarkable performance in many areas, including social networks [2, 3], molecular structures [4, 5], traffic flows [6–9], and recommendation systems [10–12]. Despite the success, their popularity in sensitive fields such as fraud detection and drug discovery [13, 14] requires an understanding of their decision-making processes. To address this challenge, some efforts have been made to explain GNN's predictions in a post-hoc manner, which aims to find a sub-graph that preserves the information about the predicted label. On top of the intuitive principle, Graph Information Bottleneck (GIB) [15, 16] maximizes the mutual information $I(G^*; Y)$ between the target label Y and the explanation G^* while constraining the size of the explanation.

However, existing methods focus on the explanation of the classification tasks, leaving another fundamental task, explainable regression, unexplored. Graph regression tasks exist widely in nowadays applications, such as predicting the molecular property [17] or traffic flow volume [18]. Explaining the instance-level predictions of graph regression is challenging due to two main obstacles. First, in the routinely adopted GIB framework, the mutual information between the explanation subgraph and label, $I(G^*; Y)$, is estimated with the Cross-Entropy between the predictions $f(G^*)$ from GNN model f and its prediction label Y. However, in the regression task, the regression label is the continuous value, making the approximation unsuitable. Another challenge is the distribution shifting problem in the usage of $f(G^*)$, the prediction of the explanation subgraph made by the GNN model f. Usually, explanation subgraphs have different topology and feature information compared to

J. Zhang et al., RegExplainer: Generating Explanations for Graph Neural Networks in Regression Tasks (Extended Abstract). Presented at the Second Learning on Graphs Conference (LoG 2023), Virtual Event, November 27–30, 2023.

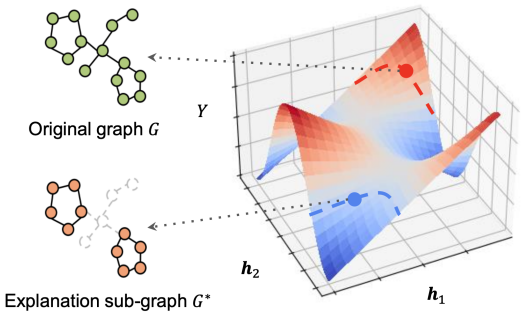

Figure 1: Intuitive illustration of the distribution shifting problem. The 3-dimensional map represents a trained GNN model f, where (h_1, h_2) represents the distribution of the graph in two dimensions, and Y represents the prediction value of the graph through f. The red and blue lines represent the distribution of the original training set and corresponding explanation sub-graph set respectively. The distribution of G^* shifts away from the original distribution, resulting in shifted prediction values.

the original graph. As a result, explanation subgraphs are out-of-distribution (OOD) of the original training graph dataset. As shown in Figure 1, a GNN model f is trained on the original graph training set and cannot be safely used to make predictions for subgraphs.

To fill the gap, in this paper, we propose RegExplainer, to generate post-hoc instance-level explanations for graph regression tasks. Specifically, we formulate a theoretical-sound objective for explainable regression based on information theory. To further address the distribution shifting issue, RegExplainer develops a new mix-up approach with contrastive learning. Our experiments show that RegExplainer provides consistent and concise explanations of GNN's predictions on regression tasks. We achieved up to 48.0% improvement when compared to the alternative baselines in our experiments. Our contributions can be summarized as follows.

• To our best knowledge, we are the first to explain GNN predictions on graph regression tasks. We addressed two challenges in explaining the graph regression task: the mutual information estimation in the GIB objective and the distribution shifting problem.

• We proposed a novel model with a new mix-up approach and contrastive learning, which could more effectively address the two challenges, and better explain the graph model on the regression tasks compared to other baselines.

• We designed two synthetic datasets, namely BA-Motif-Volume and Triangles, as well as a real-world dataset called Crippen, which could also be used in future works, to evaluate the effectiveness of our regression task explanations. Comprehensive empirical studies on both synthetic and real-world datasets demonstrate that our method can provide consistent and concise explanations for graph regression tasks.

2 Overview of Methodology

In this section, we first introduce a new objective based on GIB for explaining graph regression tasks. Then we showcase the distribution shifting problem in the GIB objective for regression and propose a novel framework through a mix-up approach to solve the distribution shifting problem.

2.1 GIB for Explaining Graph Regression

The Information Bottleneck (IB) [19, 20] provides an intuitive principle for learning dense representations that an optimal representation should contain *sufficient* information for the downstream prediction task with a *minimal* size. Based on IB, a recent work [21] unifies the most existing post-hoc explanation methods for GNN, such as GNNExplainer [22], PGExplainer [23], with the graph information bottleneck (GIB) principle [15, 16, 21]. Formally, the objective of explaining the prediction of f on G can be represented by $\arg\min_{G^*} I(G; G^*) - \alpha I(G^*; Y)$, where G is the to-be-explained original graph, G^* is the explanation sub-graph of G, Y is the original ground-truth label of G, and α is a hyper-parameter to get the trade-off between minimal and sufficient constraints. GIB uses the mutual information $I(G; G^*)$ to select the minimal explanation that inherits only the

most indicative information from G to predict the label Y by maximizing $I(G^*;Y)$, where $I(G;G^*)$ avoids imposing potentially biased constraints, such as the size or the connectivity of the selected sub-graphs [15]. Through the optimization of the sub-graph, G^* provides model interpretation. In graph classification task, a widely-adopted approximation to Eq. (3) in previous methods [22, 23] is $\arg\min_{G^*} I(G;G^*) + \alpha I(Y|G^*) \approx \arg\min_{G^*} I(G;G^*) + \alpha \text{CE}(Y,Y^*)$, where $Y^* = f(G^*)$ is the predicted label of G^* made by the to-be-explained model f, and the cross-entropy $\text{CE}(Y,Y^*)$ between the ground truth Y and Y^* is used to approximate $I(G^*;Y)$. However, the GIB objective with cross-entropy loss couldn't be applied to the regression tasks.

To address the limitation of the GIB objective in existing methods, we adopt the GIB objective with InfoNCE loss and employed the contrastive loss, which could be formally represented as

$$\arg\min_{G^*} I(G;G^*) - \alpha \mathop{\mathbb{E}}_{\mathbb{H}} \left[\log \frac{f_k\left(\boldsymbol{h}^*, \boldsymbol{h}\right)}{\frac{1}{|\mathbb{H}|}\sum_{\boldsymbol{h}_j \in \mathbb{H}} f_k\left(\boldsymbol{h}^*, \boldsymbol{h}_j\right)} \right], \tag{1}$$

where $\boldsymbol{h}$ represents the embedding of graph G and $\boldsymbol{h}^*$ represents the embedding of graph G^*. $\mathbb{E}$ is the set of instances embeddings. f_k denotes the density ratio function. A detailed description of the properties and the theoretical proofs can be found in Appendix C.

2.2 Mix-up for Distribution Shifts

We include the label of explanation sub-graph, Y^*, in our GIB objective for explaining regression. However, we argue that Y^* cannot be safely obtained due to the distribution shift problem [24, 25]. We propose a new mix-up approach to generate a mixed graph by mixing explanation sub-graph G^* with a randomly sampled label-irrelevant sub-graph G^{Δ}, which could be formally written as:

$$G_a^{(\text{mix})} = G_a^* + (G_b - G_b^*). \tag{2}$$

This equation shows that how we mix the explanation sub-graph G_a^* of G_a with the label-irrelevant sub-graph G_b^{Δ} from G_b. A detailed description of the distribution shifting problem and mix-up approach can be found in Appendix D.

2.3 Implementation

We put our overall implementation and objective function in Appendix E.

3 Experiments

In this section, we conduct experiments to demonstrate the performance of our proposed method[1]. Further experiments could be found in Appendix G:

3.1 Datasets

We formulate two synthetic datasets and a real-world dataset, as is shown in Table 1. (1) *BA-Motif-Volume*: This dataset is based on the BA-shapes [22] and modifies to summarize the node features. (2) *Triangles*: We follow the previous work [26] to construct this dataset to count the triangles. (3) *Crippen*: The Crippen dataset is a real-life dataset that was initially used to evaluate the graph regression task. The detailed formulations and configurations of our four datasets can be found in Appendix F.

3.2 Baselines

We compared the proposed RegExplainer against a comprehensive set of baselines in all datasets, including: (1) **GRAD** [22]: GRAD is a gradient-based method that learns weight vectors of edges by computing gradients of the GNN's objective function. (2) **ATT** [27]: ATT is a graph attention network (GAT) that learns attention weights for edges in the input graph. These weights can be utilized as a proxy measure of edge importance. (3) **GNNExplainer** [22]: GNNExplainer is a model-agnostic

[1] Our data and code are available at: https://anonymous.4open.science/r/RegExplainer-DDA5

Table 1: Illustration of the graph regression datasets together with the explanation faithfulness in terms of AUC-ROC on edges under four datasets on RegExplainer and other baselines. The original graph row visualizes the structure of the complete graph, the explanation row highlights the explanation sub-graph of the corresponding original graph. In the Crippen dataset, different colors of the node represent different kinds of atoms and the node feature is a one-hot vector to encode the atom type.

Dataset	BA-Motif-Volume	Triangles	Crippen
Original Graph G			
Explanation G^*			
Node Feature	Random Float Vector	Fixed Ones Vector	One-hot Vector
Regression Label	Sum of Motif Value	Count Triangles	Chemical Property Value
Explanation Type	Fix Size Sub-Graph	Dynamic Size Sub-graph	Dynamic Size Sub-graph
Explanation AUC			
GRAD	0.418 ± 0.000	0.479 ± 0.000	0.426 ± 0.000
ATT	0.512 ± 0.005	0.441 ± 0.004	0.502 ± 0.006
GNNExplainer	0.501 ± 0.009	0.500 ± 0.002	0.497 ± 0.005
PGExplainer	0.470 ± 0.057	0.511 ± 0.028	0.448 ± 0.005
RegExplainer	$\mathbf{0.758 \pm 0.177}$	$\mathbf{0.739 \pm 0.008}$	$\mathbf{0.553 \pm 0.013}$

method that learns an adjacency matrix mask by maximizing the mutual information between the predictions of the GNN and the distribution of possible sub-graph structures. (4) **PGExplainer** [23]: PGExplainer adopts a deep neural network to parameterize the generation process of explanations, which facilitates a comprehensive understanding of the predictions made by GNNs. It also produces sub-graph explanations with edge importance masks.

3.3 Quantitative Evaluation

In this section, we evaluate the performance of our approach with other baselines. For GRAD and GAT, we use the gradient-based and attention-based explanation, following the setting in the previous work [22]. We take GCN as our to-be-explained model for all post-hoc explainers. For GNNExplainer and PGExplainer, which were previously used for the classification task, we replace the Cross-Entropy loss with the MSE loss. We run and tune all the baselines on our four datasets. We evaluate the explanation from all the methods with the AUC metric, as done in the previous work. As we can see in table 1, our method achieves the best performance compared to the baselines in all four datasets.

As we can see from Table 1, RegExplainer improves the second best baseline with $0.175/34.3\%$ on average and up to $0.246/48.0\%$. The comparison between RegExplainer and other baselines indicates the advantages of our proposed approach. This improvement indicates the effectiveness of our proposed method, showing that by adding mixup and contrastive learning, we can have more faithful explanations.

4 Conclusion

We addressed the challenges in the explainability of graph regression tasks and proposed the RegExplainer, a novel method for explaining the predictions of GNNs with the post-hoc explanation sub-graph on graph regression task without requiring modification of the underlying GNN architecture or re-training. We showed how RegExplainer can leverage the distribution shifting problem and knowledge from the continuous decision boundary with the mix-up approach and the adopted GIB objective with the contrastive loss, while the problems seriously affect the performances of other explainers. We formulated four new datasets, which are BA-Motif-Volume, BA-Motif-Counting, Triangles, and Crippen for evaluating the explainers on the graph regression task, which are developed from the previous datasets and follow a similar setting. They could also benefit future studies on the XAIG-R.

References

[1] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. 1

[2] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation, 2019. 1

[3] Shengjie Min, Zhan Gao, Jing Peng, Liang Wang, Ke Qin, and Bo Fang. Stgsn — a spatial–temporal graph neural network framework for time-evolving social networks. *Knowledge-Based Systems*, 214:106746, 2021. ISSN 0950-7051. 1

[4] Hryhorii Chereda, Annalen Bleckmann, Frank Kramer, Andreas Leha, and Tim Beissbarth. Utilizing molecular network information via graph convolutional neural networks to predict metastatic event in breast cancer. In *GMDS*, pages 181–186, 2019. 1

[5] E. Mansimov, O. Mahmood, and S. Kang. Molecular geometry prediction using a deep generative graph neural network, 2019. 1

[6] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. Traffic flow prediction via spatial temporal graph neural network. In *Proceedings of The Web Conference 2020*, WWW '20, page 1082–1092, New York, NY, USA, 2020. Association for Computing Machinery. 1

[7] Mengzhang Li and Zhanxing Zhu. Spatial-temporal fusion graph neural networks for traffic flow forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):4189–4196, May 2021.

[8] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, page 1907–1913. AAAI Press, 2019. ISBN 9780999241141.

[9] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3634–3640. International Joint Conferences on Artificial Intelligence Organization, 7 2018. 1

[10] Shakila Shaikh, Sheetal Rathi, and Prachi Janrao. Recommendation system in e-commerce websites: a graph based approached. In *2017 IEEE 7th International Advance Computing Conference (IACC)*, pages 931–934. IEEE, 2017. 1

[11] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: A survey. *ACM Comput. Surv.*, 55(5), dec 2022. ISSN 0360-0300. doi: 10.1145/3535101. URL https://doi.org/10.1145/3535101.

[12] Kaige Yang and Laura Toni. Graph-based recommendation system. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 798–802. IEEE, 2018. 1

[13] Zhaoping Xiong, Dingyan Wang, Xiaohong Liu, Feisheng Zhong, Xiaozhe Wan, Xutong Li, Zhaojun Li, Xiaomin Luo, Kaixian Chen, Hualiang Jiang, et al. Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of medicinal chemistry*, 63(16):8749–8760, 2019. 1

[14] Pietro Bongini, Monica Bianchini, and Franco Scarselli. Molecular generative graph neural networks for drug discovery. *Neurocomputing*, 450:242–252, 2021. 1

[15] Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. Graph information bottleneck. *Advances in Neural Information Processing Systems*, 33:20437–20448, 2020. 1, 2, 3, 8

[16] Siqi Miao, Mia Liu, and Pan Li. Interpretable and generalizable graph learning via stochastic attention mechanism. In *International Conference on Machine Learning*, pages 15524–15543. PMLR, 2022. 1, 2, 8, 14

[17] Marc Brockschmidt. Gnn-film: Graph neural networks with feature-wise linear modulation, 2020. 1

[18] Krzysztof Rusek, Paul Almasan, José Suárez-Varela, Piotr Chołda, Pere Barlet-Ros, and Albert Cabellos-Aparicio. Fast traffic engineering by gradient descent with learned differentiable routing, 2022. 1

[19] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000. 2, 8

[20] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pages 1–5. IEEE, 2015. 2, 8

[21] Junchi Yu, Tingyang Xu, Yu Rong, Yatao Bian, Junzhou Huang, and Ran He. Graph information bottleneck for subgraph recognition. *arXiv preprint arXiv:2010.05563*, 2020. 2, 8

[22] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019. doi: 10.48550/ARXIV.1903.03894. URL https://arxiv.org/abs/1903.03894. 2, 3, 4, 8, 9, 14, 15

[23] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in neural information processing systems*, 33:19620–19631, 2020. 2, 3, 4, 8, 9, 12, 13, 14

[24] Junfeng Fang, Xiang Wang, An Zhang, Zemin Liu, Xiangnan He, and Tat-Seng Chua. Cooperative explanations of graph neural networks. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 616–624, 2023. 3, 11

[25] Jiaxing Zhang, Dongsheng Luo, and Hua Wei. Mixupexplainer: Generalizing explanations for graph neural networks with data augmentation. In *Proceedings of 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2023. 3, 11

[26] Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures?, 2020. 3, 15

[27] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. 3

[28] Caihua Shan, Yifei Shen, Yao Zhang, Xiang Li, and Dongsheng Li. Reinforcement learning enhanced explainer for graph neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 22523–22533. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/be26abe76fb5c8a4921cf9d3e865b454-Paper.pdf. 8

[29] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. In *International Conference on Machine Learning*, pages 12241–12252. PMLR, 2021. 8

[30] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. XGNN: towards model-level explanations of graph neural networks. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 430–438. ACM, 2020. 8

[31] Federico Baldassarre and Hossein Azizpour. Explainability techniques for graph convolutional networks, 2019. 8

[32] Enyan Dai and Suhang Wang. Towards self-explainable graph neural network, 2021. 8

[33] Junfeng Fang, Xiang Wang, An Zhang, Zemin Liu, Xiangnan He, and Tat-Seng Chua. *Cooperative Explanations of Graph Neural Networks*, page 616–624. Association for Computing Machinery, New York, NY, USA, 2023. ISBN 9781450394079. URL https://doi.org/10.1145/3539597.3570378. 8

[34] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM Comput. Surv.*, 50(6), dec 2017. ISSN 0360-0300. doi: 10.1145/3136625. 9

[35] Muhammed Fatih Balın, Abubakar Abid, and James Zou. Concrete autoencoders: Differentiable feature selection and reconstruction. In *International conference on machine learning*, pages 444–453. PMLR, 2019. 9

[36] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, 31, 2018. 9

[37] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019. 10

[38] Xu Liu, Yingguang Li, Qinglu Meng, and Gengxiang Chen. Deep transfer learning for conditional shift in regression. *Knowledge-Based Systems*, 227:107216, 2021. 11

[39] Yaochen Xie, Sumeet Katariya, Xianfeng Tang, Edward Huang, Nikhil Rao, Karthik Subbian, and Shuiwang Ji. Task-agnostic graph explanations, 2022. 14

[40] John S Delaney. Esol: estimating aqueous solubility directly from molecular structure. *Journal of chemical information and computer sciences*, 44(3):1000–1005, 2004. 15

[41] Scott A Wildman and Gordon M Crippen. Prediction of physicochemical parameters by atomic contributions. *Journal of chemical information and computer sciences*, 39(5):868–873, 1999. 16

[42] Benjamin Sanchez-Lengeling, Jennifer Wei, Brian Lee, Emily Reif, Peter Wang, Wesley Qian, Kevin McCloskey, Lucy Colwell, and Alexander Wiltschko. Evaluating attribution for graph neural networks. *Advances in neural information processing systems*, 33:5898–5910, 2020. 16

A Related Works

GNN Explainability. The explanation methods for GNN models could be categorized into two types based on their granularity: instance-level [22, 23, 28, 29] and model-level [30], where the former methods explain the prediction for each instance by identifying important sub-graphs, and the latter method aims to understand the global decision rules captured by the GNN. These methods could also be classified into two categories based on their methodology: self-explainable GNNs [31, 32] and post-hoc explanation methods [22, 23, 29], where the former methods provide both predictions and explanations, while the latter methods use an additional model or strategy to explain the target GNN. Additionally, CGE [33] (cooperative explanation) generates the sub-graph explanation with the sub-network simultaneously, by using cooperative learning. However, it has to treat the GNN model as a white box, which is usually unavailable in the post-hoc explanation. Existing methods haven't explored the explanation of the graph regression task and haven't considered two important challenges: the distribution shifting problem and the limitation of the GIB objective, which were addressed by our work.

GIB Objective. The Information Bottleneck (IB) [19, 20] provides an intuitive principle for learning dense representations that an optimal representation should contain *sufficient* information for the downstream prediction task with a *minimal* size. Based on IB, a recent work [21] unifies the most existing post-hoc explanation methods for GNN, such as GNNExplainer [22], PGExplainer [23], with the graph information bottleneck (GIB) principle [15, 16, 21]. Formally, the objective of explaining the prediction of f on G can be represented by

$$\arg\min_{G^*} I(G; G^*) - \alpha I(G^*; Y), \tag{3}$$

where G is the to-be-explained original graph, G^* is the explanation sub-graph of G, Y is the original ground-truth label of G, and α is a hyper-parameter to get the trade-off between minimal and sufficient constraints. GIB uses the mutual information $I(G; G^*)$ to select the minimal explanation that inherits only the most indicative information from G to predict the label Y by maximizing $I(G^*; Y)$, where $I(G; G^*)$ avoids imposing potentially biased constraints, such as the size or the connectivity of the selected sub-graphs [15]. Through the optimization of the sub-graph, G^* provides model interpretation. In graph classification task, a widely-adopted approximation to Eq. (3) in previous methods [22, 23] is:

$$\begin{aligned} \arg\min_{G^*} I(G; G^*) + \alpha I(Y|G^*) \approx \\ \arg\min_{G^*} I(G; G^*) + \alpha \text{CE}(Y, Y^*), \end{aligned} \tag{4}$$

where $Y^* = f(G^*)$ is the predicted label of G^* made by the to-be-explained model f, and the cross-entropy $\text{CE}(Y, Y^*)$ between the ground truth Y and Y^* is used to approximate $I(G^*; Y)$. The approximation is based on the definition of mutual information $I(G^*; Y) = H(Y) - H(Y|G^*)$: with entropy $H(Y)$ being static and independent of the explanation process, minimizing the mutual information between the explanation sub-graph G^* and Y can be reformulated as maximizing the conditional entropy of Y given G^*, which can be approximated by $\text{CE}(Y, Y^*)$.

B Preliminary

Notation and Problem Formulation. We use $G = (\mathcal{V}, \mathcal{E}; \boldsymbol{X}, \boldsymbol{A})$ to represent a graph, where $\mathcal{V}$ equals to $\{v_1, v_2, ..., v_n\}$ represents a set of n nodes and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ represents the edge set. Each graph has a feature matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ for the nodes, wherein $\boldsymbol{X}$, $X_i \in \mathbb{R}^{1 \times d}$ is the d-dimensional node feature of node v_i. $\mathcal{E}$ is described by an adjacency matrix $\boldsymbol{A} \in \{0, 1\}^{n \times n}$, where $A_{ij} = 1$ means that there is an edge between node v_i and v_j; otherwise, $A_{ij} = 0$. For the graph prediction task, each graph G_k has a label $Y_k \in \mathcal{C}$, where $k \in \{1, ..., N\}$, $\mathcal{C}$ is the set of the classification categories or regression values, with a GNN model f trained to make the prediction, i.e., $f : (\boldsymbol{X}, \boldsymbol{A}) \mapsto \mathcal{C}$.

Problem 1 (Post-hoc Instance-level GNN Explanation)**.** *Given a trained GNN model f, for an arbitrary input graph $G = (\mathcal{V}, \mathcal{E}; \boldsymbol{X}, \boldsymbol{A})$, the goal of post-hoc instance-level GNN explanation is to find a sub-graph G^* that can explain the prediction of f on G.*

In non-graph structured data, the informative feature selection has been well studied [34], as well as in traditional methods, such as concrete auto-encoder [35], which can be directly extended to explain features in GNNs. In this paper, we focus on discovering the important sub-graph typologies following the previous work [22, 23]. Formally, the obtained explanation G^* is depicted by a binary mask $\boldsymbol{M}^* \in \{0,1\}^{n\times n}$ on the adjacency matrix, e.g., $G^* = (\mathcal{V}, \mathcal{E}; \boldsymbol{X}, \boldsymbol{A} \odot \boldsymbol{M}^*)$, $\odot$ means elements-wise multiplication. The mask highlights components of G which are essential for f to make the prediction.

C GIB for Explaining Graph Regression

As introduced in Section A, in the classification task, $I(G^*; Y)$ in Eq. (3) is commonly approximated by cross-entropy $\text{CE}(Y^*, Y)$ [36]. However, it is non-trivial to extend it for regression tasks because Y is a continuous variable and it is intractable to compute the cross-entropy $\text{CE}(Y^*, Y)$ or the mutual information $I(G^*; Y)$, where G^* is a graph variable with a continuous variable Y^* as its label.

C.1 Optimizing the Lower Bound of $I(G^*; Y)$

To address the challenge of computing the mutual information $I(G^*; Y)$ with a continuous Y, we propose a novel objective for explaining graph regression.

Instead of minimizing $I(G^*; Y)$ directly, we propose to maximize a lower bound for the mutual information by including the label of G^*, denoted by Y^*, and approximate $I(G^*; Y)$ with $I(Y^*; Y)$, where Y^* is the prediction label of G^*:

$$\arg\min_{G^*} I(G; G^*) - \alpha I(Y^*; Y). \tag{5}$$

$I(Y^*; Y)$ has the following property, upon with we can approximate Eq. (3) with Eq. (5):

Property 1. *$I(Y^*; Y)$ is a lower bound of $I(G^*; Y)$.*

Proof. From the definition of Y^*, we can make a safe assumption that there is a many-to-one map (function), denoted by h, from G^* to Y^* as Y^* is the prediction label for G^*. For simplicity, we assume a finite number of explanation instances for each label y^*, and each explanation instance, denoted by g^*, is generated independently. Then, we have $p(y^*) = \sum_{g^* \in \mathbb{G}(y^*)} p(g^*)$, where $\mathbb{G}(y^*) = \{g | h(g) = y^*\}$ is the set of explanations whose labels are y^*.

Based on the definition of mutual information, we have:

$$\begin{aligned}
I(G^*; Y) &= \int_y \int_{g^*} p_{(G^*,Y)}(g^*, y) \log \frac{p_{(G^*,Y)}(g^*, y)}{p_{G^*}(g^*) p_Y(y)} d_{g^*} d_y \\
&= \int_y \int_{g^*} p_{(G^*,Y^*,Y)}(g^*, h(g^*), y) \\
&\qquad \log \frac{p_{(G^*,Y^*,Y)}(g^*, h(g^*), y)}{p_{G^*}(g^*) p_Y(y)} d_{g^*} d_y \\
&= \int_y \int_{g^*} p_{(G^*,Y^*,Y)}(g^*, h(g^*), y) \\
&\qquad \log \frac{p_{(G^*,Y^*,Y)}(g^*, h(g^*), y)}{p_{(G^*,Y^*)}(g^*, h(g^*)) p_Y(y)} d_{g^*} d_y \\
&= \int_y \int_{y^*} \sum_{g^* \in \mathbb{G}(y^*)} p_{(G^*,Y^*,Y)}(g^*, y^*, y) \\
&\qquad \log \frac{p_{(G^*,Y^*,Y)}(g^*, y^*, y)}{p_{(G^*,Y^*)}(g^*, y^*) p_Y(y)} d_{y^*} d_y
\end{aligned}$$

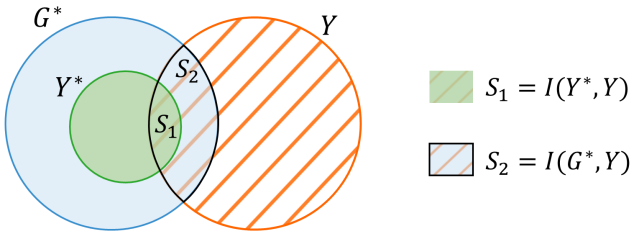

Figure 2: Intuitive illustration about why $I(G^*;Y) \geq I(Y^*;Y)$. G^* contains more mutual information as having more overlapping area with Y than the overlapping area between Y^* and Y.

Based on our many-to-one assumption , while each g^* is generated independently , we know that if $g \notin \mathbb{G}(y^*)$, then $p_{(G^*,Y^*,Y)}(g^*,y^*,y) = 0$. Thus, we have:

$$
\begin{aligned}
I(G^*;Y) &= I(G^*;Y) \\
&+ \int_y \int_{y^*} \sum_{g \notin \mathbb{G}(y^*)} p_{(G^*,Y^*,Y)}(g^*,y^*,y) \\
&\qquad \log \frac{p_{(G^*,Y^*,Y)}(g^*,y^*,y)}{p_{(G^*,Y^*)}(g^*,y^*)p_Y(y)} d_{y^*} d_y \\
&= \int_y \int_{y^*} \int_{g^*} p_{(G^*,Y^*,Y)}(g^*,y^*,y) \\
&\qquad \log \frac{p_{(G^*,Y^*,Y)}(g^*,y^*,y)}{p_{(G^*,Y^*)}(g^*,y^*)p_Y(y)} d_{g^*} d_{y^*} d_y \\
&= I(G^*,Y^*;Y).
\end{aligned}
$$

With the chain rule for mutual information, we have $I(G^*,Y^*;Y) = I(Y^*;Y) + I(G^*;Y|Y^*)$. Then due to the non-negativity of the mutual information, we have $I(G^*,Y^*;Y) \geq I(Y^*;Y)$. □

Intuitively, the property of $I(Y^*;Y)$ is guaranteed by the chain rule for mutual information and the independence between each explanation instance g^*. An intuitive demonstration is shown in Figure 2.

C.2 Estimating $I(Y^*;Y)$ with InfoNCE

Now the challenge becomes the estimation of the mutual information $I(Y^*;Y)$. Inspired by the model of Contrastive Predictive Coding [37], in which InfoNCE loss is interpreted as a mutual information estimator, we further extend the method so that it could apply to InfoNCE loss in explaining graph regression. In our graph explanation scenario, the InfoNCE Loss defined in Eq. (6) could also be utilized as a lower bound of $I(Y^*;Y)$, as shown in the following property with proofs:

Property 2. InfoNCE Loss is a lower bound of the $I(Y^*;Y)$:

$$
I(Y^*;Y) \geq \mathop{\mathbb{E}}_{\mathbb{Y}} \left[\log \frac{f_k(Y^*,Y)}{\frac{1}{|\mathbb{Y}|} \sum_{Y_j \in \mathbb{Y}} f_k(Y^*,Y_j)} \right], \tag{6}
$$

where Y_j is randomly sampled graph neighbors, $\mathbb{Y}$ is the set of the neighbors, and f_k is the density ratio function.

Proof. As in the InfoNCE method, the mutual information between Y^* and Y is defined as:

$$
I(Y^*;Y) = \sum_{Y^*,Y} p(Y^*,Y) \log \frac{p(Y|Y^*)}{P(Y)} \tag{7}
$$

However, the ground truth joint distribution $p(Y^*,Y)$ is not controllable, so, we turn to maximize the density ratio

$$f_k\left(Y^*, Y\right) \propto \frac{p(Y|Y^*)}{p(Y)} \tag{8}$$

We want to put the representation function of mutual information into the NCE Loss

$$\mathcal{L}_N = -\mathop{\mathbb{E}}_{\mathbb{Y}} \log \left[\frac{f_k\left(Y^*, Y\right)}{\sum_{Y' \in \mathbb{Y}} (Y^*, Y')} \right], \tag{9}$$

where $\mathcal{L}_N$ denotes the NCE loss and by inserting the optimal $f_k\left(Y^*, Y\right)$ into Eq. (9), we could get:

$$\begin{aligned}
\mathcal{L}_{\text{contr}} &= -\mathop{\mathbb{E}}_{\mathbb{Y}} \log \left[\frac{\frac{p(Y|Y^*)}{p(Y)}}{\frac{p(Y|Y^*)}{p(Y)} + \sum_{Y' \in \mathbb{Y}_{\text{neg}}} \frac{p(Y^*, Y')}{p(Y')}} \right] \\
&= \mathop{\mathbb{E}}_{\mathbb{Y}} \log \left[1 + \frac{p(Y|Y^*)}{p(Y)} \sum_{Y' \in \mathbb{Y}_{\text{neg}}} \frac{p(Y^*, Y')}{p(Y')} \right] \\
&\approx \mathop{\mathbb{E}}_{\mathbb{Y}} \log \left[1 + \frac{p(Y|Y^*)}{p(Y)} (N-1) \mathop{\mathbb{E}}_{Y'} \frac{p(Y^*, Y')}{p(Y')} \right] \\
&= \mathop{\mathbb{E}}_{\mathbb{Y}} \log \left[1 + \frac{p(Y|Y^*)}{p(Y)} (N-1) \right] \\
&\geq \mathop{\mathbb{E}}_{\mathbb{Y}} \log \left[\frac{p(Y|Y^*)}{p(Y)} N \right] \\
&= -I(Y^*, Y) + \log(N)
\end{aligned} \tag{10}$$

□

To employ the contrastive loss, we use the representation embedding to approximate Y, where $\boldsymbol{h}^*$ represents the embedding for G^* and $\boldsymbol{h}$ represents the embedding for G. We use $\mathbb{H}$ to represent the neighbors set $\mathbb{Y}$ accordingly. Thus, we approximate Eq. (5) as:

$$\arg\min_{G^*} I(G; G^*) - \alpha \mathop{\mathbb{E}}_{\mathbb{H}} \left[\log \frac{f_k\left(\boldsymbol{h}^*, \boldsymbol{h}\right)}{\frac{1}{|\mathbb{H}|} \sum_{\boldsymbol{h}_j \in \mathbb{H}} f_k\left(\boldsymbol{h}^*, \boldsymbol{h}_j\right)} \right] \tag{11}$$

D Mix-up for Distribution Shifts

In the above section, we include the label of explanation sub-graph, Y^*, in our GIB objective for explaining regression. However, we argue that Y^* cannot be safely obtained due to the distribution shift problem [24, 25].

D.1 Distribution Shifts in GIB for Regression

Suppose that the to-be-explained model f is trained on a dataset $\{(G_k, Y_k)\}_{k=1}^N$. Usually in supervised learning (without domain adaptation), we suppose that the examples (G_k, Y_k) are drawn i.i.d. from a distribution $\mathcal{D}_{\text{train}}$ of support $G \times Y$ (unknown and fixed). The objective is then to learn f such that it commits the least error possible for labeling new examples coming from the distribution $\mathcal{D}_{\text{train}}$. However, as pointed out in previous studies, there is a shift between the distribution explanation sub-graphs, denoted by $\mathcal{D}_{\text{exp}}$ and $\mathcal{D}_{\text{train}}$. As the explanation, sub-graphs tend to be small and dense. The distribution shift problem is severe in regression problems due to the continuous decision boundary [38].

Figure 3 shows the existence of distribution shifts between $f(G^*)$ and Y in graph regression tasks. For each dataset, we sort the indices of the data samples according to the value of their labels, and visualize the label Y, prediction $f(G)$ of the original graph from the trained GNN model f, and prediction $f(G^*)$ of the explanation sub-graph G^* from f. As we can see in Figure 3, in all four graph regression datasets, the red points are well distributed around the ground-truth blue points,

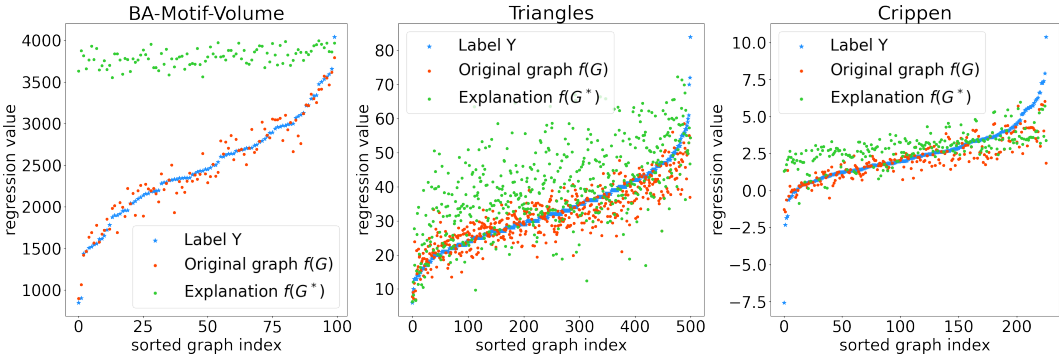

Figure 3: Visualization of distribution shifting problem on three graph regression datasets. The points represent the regression value, where the blue points mean label Y, red points mean prediction $f(G)$ of the original graph, and the green points mean prediction $f(G^*)$ of explanations on the four datasets, the x-axis is the indices of the graph, sorted by the value of the label Y.

indicating that $f(G)$ is close to Y. In comparison, the green points shift away from the red points, indicating the shifts between $f(G^*)$ and $f(G)$. Intuitively, this phenomenon indicates the GNN model f could make correct predictions only with the original graph G yet could not predict the explanation sub-graph G^* correctly. This is because the GNN model f is trained with the original graph sets, whereas the explanation G^* as the sub-graph is out of the distribution from the original graph sets. With the shift between $f(G)$ and $f(G^*)$, the optimal solution in Eq. (5) is unlikely to be the optimal solution for Eq. (3).

D.2 Graph Mix-up Approach.

To address the distribution shifting issue between $f(G)$ and $f(G^*)$ in the GIB objective, we introduce the mix-up approach to reconstruct a within-distribution graph, $G^{(\text{mix})}$, from the explanation graph G^*. We follow [23] to make a widely-accepted assumption that a graph can be divided by $G = G^* + G^{\Delta}$, where G^* presents the underlying sub-graph that makes important contributions to GNN's predictions, which is the expected explanatory graph, and G^{Δ} consists of the remaining label-independent edges for predictions made by the GNN. Both G^* and G^{Δ} influence the distribution of G. Therefore, we need a graph $G^{(\text{mix})}$ that contains both G^* and G^{Δ}, upon which we use the prediction of $G^{(\text{mix})}$ made by f to approximate Y^* and $\boldsymbol{h}^*$.

Specifically, for a target graph G_a in the original graph set to be explained, we generate the explanation sub-graph $G_a^* = G_a - G_a^{\Delta}$ from the explainer. To generate a graph in the same distribution of original G_a, we can randomly sample a graph G_b from the original set, generate the explanation sub-graph of G_b^* with the same explainer and retrieve its label-irrelevant graph $G_b^{\Delta} = G_b - G_b^*$. Then we could merge G_a^* together with G_b^{Δ} and produce the mix-up explanation $G_a^{(\text{mix})}$. Formally, we can have $G_a^{(\text{mix})} = G_a^* + (G_b - G_b^*)$.

Since we are using the edge weights mask to describe the explanation, we can denote G_a and G_b with the adjacency matrices $\boldsymbol{A}_a$ and $\boldsymbol{A}_b$, their edge weight mask matrices as $\boldsymbol{M}_a$ and $\boldsymbol{M}_b$. If G_a and G_b are aligned graphs with the same number of nodes, we could simply mix them up by $\boldsymbol{M}_a^{(\text{mix})} = \boldsymbol{M}_a^* + (\boldsymbol{I}_b - \boldsymbol{M}_b^*)$, where $\boldsymbol{M}$ denotes the weight of the adjacency matrix and $\boldsymbol{I}_b$ denotes the zero-ones matrix as weights of all edges in the adjacency matrix of G_b, where 1 represents the existing edge and 0 represents there is no edge between the node pair.

If G_a and G_b are not aligned with the same number of nodes, we can use a connection adjacency matrix $\boldsymbol{A}_{\text{conn}}$ and mask matrix $\boldsymbol{M}_{\text{conn}}$ to merge two graphs with different numbers of nodes. Specifically, the mix-up adjacency matrix could be formed as:

$$\boldsymbol{A}_a^{(\text{mix})} = \begin{bmatrix} \boldsymbol{A}_a & \boldsymbol{A}_{\text{conn}} \\ \boldsymbol{A}_{\text{conn}}^T & \boldsymbol{A}_b \end{bmatrix}. \tag{12}$$

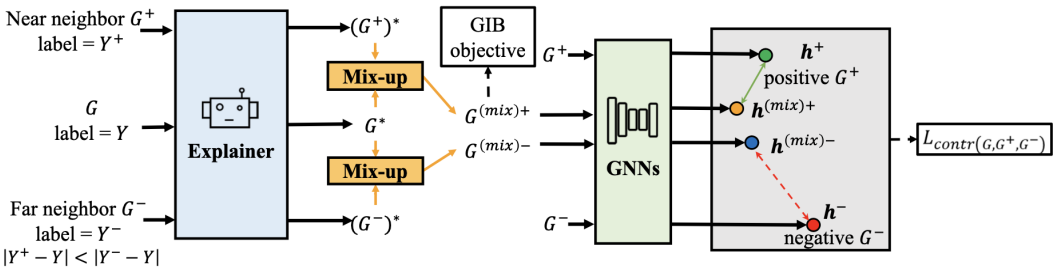

Figure 4: Illustration of RegExplainer. G is the to-be-explained graph, G^+ is the randomly sampled positive graph and G^- is the randomly sampled negative graph. The explanation of the graph is produced by the explainer model. Then graph G is mixed with G^+ and G^- respectively to produce $G^{(\text{mix})+}$ and $G^{(\text{mix})-}$. Then the graphs are fed into the trained GNN model to retrieve the embedding vectors $\boldsymbol{h}^+$, $\boldsymbol{h}^-$, $\boldsymbol{h}^{(\text{mix})+}$ and $\boldsymbol{h}^{(\text{mix})-}$. We use contrastive loss to minimize the distance between $G^{(\text{mix})+}$ and the positive sample and maximum the distance between $G^{(\text{mix})-}$ and the negative sample. The explainer is trained with the GIB objective and contrastive loss.

And the mix-up mask matrix could be formed as:

$$\boldsymbol{M}_a^{(\text{mix})} = \begin{bmatrix} \boldsymbol{M}_a^* & \boldsymbol{M}_{\text{conn}} \\ \boldsymbol{M}_{\text{conn}}^T & \boldsymbol{M}_b^\Delta \end{bmatrix} \tag{13}$$

Finally, we could form $G_a^{(\text{mix})}$ as $(\boldsymbol{X}^{(\text{mix})}, \boldsymbol{A}_a^{(\text{mix})} \odot \boldsymbol{M}_a^{(\text{mix})})$, where $\boldsymbol{X}^{(\text{mix})} = [\boldsymbol{X}_a; \boldsymbol{X}_b]$. The detailed algorithm for mix-up is shown in Algorithm 1.

Algorithm 1 Graph Mix-up Algorithm

Input: Target to-be-explained graph $G_a = (\boldsymbol{X}_a, \boldsymbol{A}_a)$, G_b sampled from a set of graphs $\mathbb{G}$, the number of random connections η, explainer model E.
Output: Graph $G^{(\text{mix})}$.

```
Generate mask matrix M_a = E(G_a)
Generate mask matrix M_b = E(G_b)
Sample η random connections between G_a and G_b as A_conn
Mix-up adjacency matrix A_a^(mix) with Eq. (12)
Mix-up edge mask M_a^(mix) with Eq. (13)
Mix-up node features X^(mix) = [X_a; X_b]
return G^(mix) = (X^(mix), A_a^(mix) ⊙ M_a^(mix))
```

We show that this mix-up approach has the following property, with its proof:

Property 3. $G^{(\text{mix})}$ is within the distribution of $\mathcal{D}_{\text{train}}$.

Proof. Following the previous work [23], we denote a graph $G = G^* + G^\Delta$, where G^* is the sub-graph explanation and G^Δ is the label-irrelevant graph. A common acknowledgment is that for a graph G with label Y, the explanation G^* holds the label-preserving information, which is the important sub-graph, while G^Δ also holds useful information which makes sure connecting it with G^* would maintain the distribution of the graph and not lead to another label. We denote the distribution for the graphs as $G \sim \mathcal{D}_{\text{train}} = \mathbb{P}_{\mathcal{G}}$, $G^* \sim \mathbb{P}_{\mathcal{G}^*}$, $G^\Delta \sim \mathbb{P}_{\mathcal{G}^\Delta}$, where $\mathcal{D}_{\text{train}}$ means the distribution of train dataset. When we produce $G^{(\text{mix})}$, we independently sample a label-irrelevant graph G^Δ and mix it up with target explanation G^*. So, we could write the distribution of $G^{(\text{mix})}$ as :

$$G^{(\text{mix})} \sim \mathbb{P}_{\mathcal{G}^*} * \mathbb{P}_{\mathcal{G}^\Delta} = \mathbb{P}_{(\mathcal{G}^* + \mathcal{G}^\Delta)} = \mathbb{P}_{\mathcal{G}} = \mathcal{D}_{\text{train}} \tag{14}$$

Thus, we prove that $G^{(\text{mix})}$ is within the distribution of $\mathcal{D}_{\text{train}}$. □

E Implementation

E.1 InfoNCE Loss

After generating the mix-up explanation $G^{(\text{mix})}$, we specify the contrastive loss to further train the parameterized explainer with a triplet of graphs $\langle G, G^+, G^- \rangle$ as the implementation of InfoNCE Loss in Eq. (11). Intuitively, for each target graph G with label Y to be explained, we can define two randomly sampled graphs as positive graph G^+ and negative instance G^- where G^+'s label, Y^+ is closer to Y than G^-'s label, Y^-, i.e., $|Y^+ - Y| < |Y^- - Y|$. Therefore, the distance between the distributions of the positive pair $\langle G, G^+ \rangle$ should be smaller than the distance between the distributions of the negative pair $\langle G, G^- \rangle$.

In practice, G^+ and G^- are randomly sampled from the graph dataset, upon which we calculate their similarity score with the target graph G. The sample with a higher score would be the positive sample and the other one would be the negative sample. Specifically, we use $\text{sim}(\boldsymbol{h}, \boldsymbol{h}_j) = \boldsymbol{h}^T \boldsymbol{h}_j$ to compute the similarity score, where G_j could be G^+ or G^-. $\boldsymbol{h}$ is generated by feeding G into the GNN model f and retrieving the embedding vector before the dense layers.

Learning through the triplet instances could effectively reinforce the ability of the explainer to learn the explanation self-supervised. A similar idea goes with mix-up graphs which we propose to address the distribution shifts in GIB. After we mix up target graph G with the neighbors G^+ and G^- and get $G^{(\text{mix})+}$ and $G^{(\text{mix})-}$ respectively, the distance between $G^{(\text{mix})+}$ and G^+ should be smaller than the distance between $G^{(\text{mix})-}$ and G^-. For example, given graph G with label Y, we can randomly sample a near neighbor G^+ with label Y^+ and a far neighbor G^- with label Y^-, where $|Y^- - Y| > |Y^+ - Y|$. With an explainer, we can get the explanation of G as G^*, and the label-independent parts of G^+ and G^-. Then if we mix G^* with these label-independent parts, we could have $G^{(\text{mix})+}$ and $G^{(\text{mix})-}$. Since we only mix the label-independent parts with G^*, the prediction $f(G^{(\text{mix})+})$ and $f(G^{(\text{mix})-})$ should be similar to $f(G)$. Since $|Y^- - Y| > |Y^+ - Y|$, we could have $|f(G^-) - f(G^{(\text{mix})-})| > |f(G^+) - f(G^{(\text{mix})+})|$, where $f(G)$ represents the prediction label of graph G.

Formally, given a target graph G, the sampled positive graph G^+ and negative graph G^-, we formulate the contrastive loss in Eq. (11) as the following:

$$\mathcal{L}_{\text{contr}}(G, G^+, G^-) = \\ -\log \frac{\exp((\boldsymbol{h}^{(\text{mix})+})^T \boldsymbol{h}^+)}{\exp((\boldsymbol{h}^{(\text{mix})+})^T \boldsymbol{h}^+) + \exp((\boldsymbol{h}^{(\text{mix})-})^T \boldsymbol{h}^-)} \tag{15}$$

where $\exp()$ function is used to instantiate the density ratio function f_k, the denominator is a sum over the ratios of both positive and negative samples in the triplet $\mathbb{Y}$.

E.2 Size Constraints

We optimize $I(G; G^*)$ in Eq. (11) to constraint the size of the explanation sub-graph G^*. The upper bound of $I(G; G^*)$ is optimized as the estimation of the KL-divergence between the probabilistic distribution between the G^* and G, where the KL-divergence term can be divided into two parts as the entropy loss and size loss [16]. In practice, we follow the previous work [22, 23, 39] to implement them. Specifically,

$$\mathcal{L}_{\text{size}}(G, G^*) = \gamma \sum_{(i,j) \in \mathcal{E}} (M^*_{ij}) - \log \sigma(\boldsymbol{h}^* (\boldsymbol{h}^*)^T), \tag{16}$$

where $\sum_{(i,j) \in \mathcal{E}} (M^*_{ij})$ means sum the weights of the existing edges in the edge weight mask $\boldsymbol{M}^*$ for the explanation G^*; $\boldsymbol{h}^*$ is extracted from the embedding of the graph G^* before the GNN model f transforming it into prediction Y^*, σ means the sigmoid function and γ is the weight for the size of the masked graph.

E.3 Overall Objective Function

In practice, the denominator in Eq. (11) works as a regularization to avoid trivial solutions. Since the label Y is given and independent of the optimization process, we could also employ the MSE

loss between Y^* and Y additionally, regarding InfoNCE loss only estimates the mutual information between the embeddings. Formally, the overall loss function could be implemented as:

$$\mathcal{L}_{\text{GIB}} = \mathcal{L}_{\text{size}}(G, G^*) - \alpha \mathcal{L}_{\text{contr}}(G, G^+, G^-) \tag{17}$$

$$\mathcal{L} = \mathcal{L}_{\text{GIB}} + \beta \mathcal{L}_{\text{MSE}}(f(G), f(G^{(\text{mix})+})), \tag{18}$$

where $G^{(\text{mix})+}$ means mix G^* with the positive sample G^+ and the hyper-parameters are α and β.

Algorithm 2 Training Explainer

Input: A set of graphs $\mathbb{G}$, trained GNN model f, explainer model E.
Output: Trained explainer E.

```
Initialize explainer model E.
for e ∈ epochs do
    for G ∈ 𝔾 do
        G_b, G_c ← Randomly sample two graphs from 𝔾
        G^+, G^− ← Compare similarity(G_b, G_c) to G
        G^(mix)+ ← Mix-up (G, G^+)
        G^(mix)− ← Mix-up (G, G^−)
        Compute L_contr(G, G^+, G^−) with Eq. (15)
        Compute L_GIB with Eq. (17)
        Compute overall loss L with Eq. (18)
    end for
    Update E with back propagation.
end for
return Explainer E
```

Algorithm 2 shows the training procedure for our explainer. For each epoch and each to-be-explained graph G, we first randomly sample two neighbors and decide the near neighbor G^+ and far neighbor G^- according to the similarity between their embedding vectors and $\boldsymbol{h}$ of G respectively, with the graph with higher similarity to G being the near neighbor G^+. We generate the explanation for graphs and mix G with G^+ and G^- respectively. We calculate the contrastive loss for triplet $\langle G, G^+, G^- \rangle$ with Eq. (15) and the GIB loss, which contains the size loss and contrastive loss. We also calculate the MSE loss between $f(G^{(\text{mix})+})$ and $f(G)$. The overall loss is the sum of GIB loss and MSE loss. We update the trainable parameters in the explainer with the overall loss.

F Datasets

Here is a detailed description of our four datasets:

(1) *BA-Motif-Volume*: This dataset is based on the BA-shapes [22] and makes a modification, which is adding random float values from [0.00, 100.00] as the node feature. We then sum the node values on the motif as the regression label of the whole graph, which means the GNNs should recognize the [house] motif and then sum features to make the prediction.

(2) *BA-Motif-Counting*: Different from BA-Motif-Volume, where node features are summarized, in this dataset, we attach various numbers of motifs to the base BA random graph and pad all graphs to equal size. The number of motifs is counted as the regression label. Padding graphs to the same size could prevent the GNNs from making trivial predictions based on the total number of nodes.

(3) *Triangles*: We follow the previous work [26] to construct this dataset. The dataset is a set of 5000 Erdős–Rényi random graphs denoted as $ER(m, p)$, where $m = 30$ is the number of nodes in each graph and $p = 0.2$ is the probability for an edge to exist. The size of 5000 was chosen to match the previous work. The regression label for this dataset is the number of triangles in a graph and GNNs are trained to count the triangles.

(4) *Crippen*: The Crippen dataset is a real-life dataset that was initially used to evaluate the graph regression task. The dataset has 1127 graphs reported in the Delaney solubility dataset [40] and has

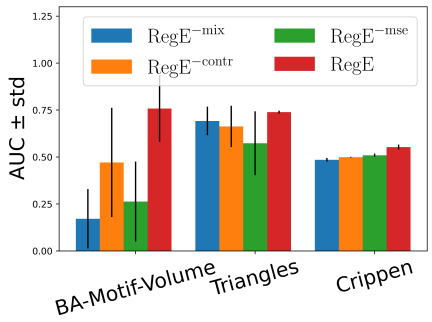

Figure 5: Ablation study of RegExplainer. We evaluated the performance of original RegExplainer and its variants that exclude mix-up approach, contrastive loss, or MSE loss respectively. The black solid line shows the standard deviation value.

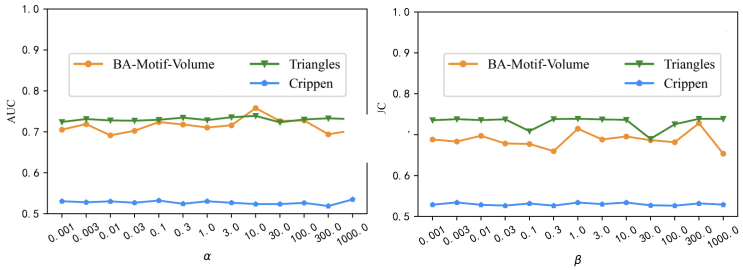

Figure 6: Hyper-parameters study of α and β on four datasets with RegExplainer. In both figures, the x-axis is the value of different hyper-parameter settings and the y-axis is the value of the average AUC score over ten runs with different random seeds.

weights of each node assigned by the Crippen model [41], which is an empirical chemistry model predicting the water-actual partition coefficient. We adopt this dataset, firstly shown in the previous work [42], and construct edge weights by taking the average of the two connected nodes' weights.

G Additional Experimants

G.1 Ablation Study

We conducted an ablation study to show how our proposed components, specifically, the mix-up approach and contrastive learning, contribute to the final performance of RegExplainer. To this end, we denote RegExplainer as RegE and design three types of variants as follows: (1) RegE$^{-\text{mix}}$: We remove the mix-up processing after generating the explanations and feed the sub-graph G^* into GIB objective and contrastive loss directly. (2) RegE$^{-\text{contr}}$: We remove the contrastive loss term but still maintain the mix-up processing and MSE loss. (3) RegE$^{-\text{mse}}$: We remove the MSE loss computation item from the objective function.

Additionally, we set all variants with the same configurations as original RegExplainer, including learning rate, training epochs, and hyper-parameters η, α, and β. We trained them on all four datasets and conduct the results in Figure 5. We observed that the proposed RegExplainer outperforms its variants in all datasets, which indicates that each component is necessary and the combination of them is effective.

G.2 Hyper-parameter Sensitivity Study

In this section, we investigate the hyper-parameters of our approach, which include α and β, across all four datasets. The hyper-parameter α controls the weight of the contrastive loss in the GIB objective while the β controls the weight of the MSE loss. We determined the optimal values of α and β with grid search. The experimental results can be found in Figure 6. Our findings indicate that

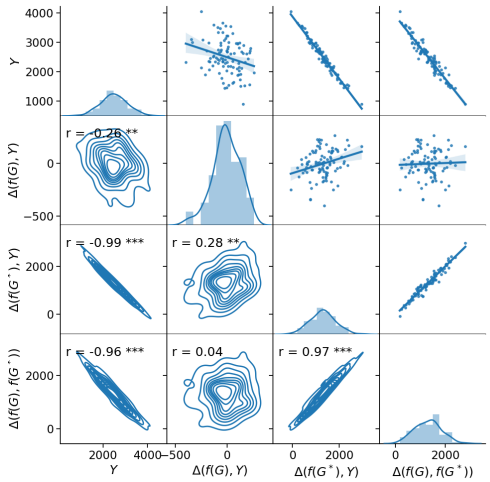

Figure 7: Correlations between the predictions and their shifting on BA-Motif-Volume. The value of r indicates the Pearson Correlation Coefficient, and the values with * indicate statistical significance for correlation, where *** indicates the p-value for testing non-correlation $p \leq 0.001$. Each point represents one graph instance.

Table 2: Prediction shifting study on the RMSE of $(f(G), Y)$, $(f(G^*), Y)$, $(f(G), f(G^*))$ respectively.

Dataset	$(f(G), Y)$	$(f(G^*), Y)$	$(f(G), f(G^*))$
BA-Motif-Volume	131.42	1432.07	1427.07
Triangles	5.28	12.38	12.40
Crippen	1.13	1.54	1.17

our approach, RegExplainer, is stable and robust when using different hyper-parameter settings, as evidenced by consistent performance across a range.

We also have additional experimental results in the Appendix on how the distribution shifts exist in regression tasks and how our proposed method alleviates the distribution shifts.

G.3 Alleviating Distribution Shifts

In this section, we visualize the regression values of the graphs and calculate the prediction shifting distance for each dataset and analyze their correlations to the distance of the decision boundaries. We put our results into Figure 7 and Table 2.

We observed that in Figure 3, red points distribute surround the blue points but green points are shifted away, which indicates that the explanation sub-graph couldn't help GNNs make correct predictions. As shown in Table 2, we calculate the RMSE score between [the] $f(G)$ and Y, $f(G^*)$ and Y, $f(G)$ and $f(G^*)$ respectively, where $f(G)$ is the prediction the original graph, $f(G^*$ is the prediction of the explanation sub-graph, and Y is the regression label. We could observe that $f(G^*)$ shows a significant prediction shifting from $f(G)$ and Y, indicating that the mutual information calculated with the original GIB objective Eq. (4) would be biased.

We further explore the relationship of the prediction shifting against the label value with dataset BA-Motif-Volume, which represents the semantic decision boundary. In Figure 7, each point represents a graph instance, where Y represents the ground-truth label, and Δ represents the absolute value difference. It's clear that both the $\Delta(f(G^*), Y)$ and $\Delta(f(G), f(G^*))$ strongly correlated to Y with statistical significance, indicating the prediction shifting problem is related to the continuous ordered decision boundary, which is present in regression tasks.